

MENU | **SEARCH** | **INDEX** | **DETAIL** | **JAPANESE** | **LEGAL STATUS**

1 / 1

PATENT ABSTRACTS OF JAPAN

(11) Publication number : 2002-259146
 (43) Date of publication of application : 13.09.2002

(51) Int. Cl. G06F 9/46
 G06F 12/00

(21) Application number : 2001-145592 (71) Applicant : MATSUSHITA ELECTRIC IND CO LTD
 (22) Date of filing : 15.05.2001 (72) Inventor : SHIOMI RYUICHI
 HAYAMA SATORU
 HIRAMOTO KENJI
 KUBOOKA YUUKO
 DOI SHIGENORI

(30) Priority

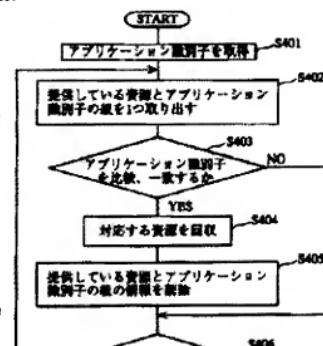
Priority number :	2000141492	Priority date :	15.05.2000	Priority country :	JP
	2000141493 :		15.05.2000 :		JP
	2000398746		27.12.2000		JP

(54) DEVICE AND METHOD FOR EXECUTING APPLICATION

(57) Abstract:

PROBLEM TO BE SOLVED: To provide an application execution device capable of recovering resources without using kernels and a Java application execution device capable of continuously executing Java applications without restarting Java middleware and reducing the load of garbage collection processing.

SOLUTION: In providing a resource at the request for providing a resource from an application, a library part acquires an identifier (ID) for identifying the application requesting for provision of the resource and stores a pair of the provided resource and the acquired ID in a table.



(19)日本特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2002-259146

(P2002-259146A)

(43)公開日 平成14年9月13日(2002.9.13)

(51)Int.Cl.
G 06 F
9/46
12/00

並用記号
3 4 0
6 9 1

F I
G 06 F
9/46
12/00

5 4 0 F
5 B 0 0 0
5 9 1
5 B 0 9 8

ターボーI(参考)

審査請求 実用求 請求項の範囲 O L (全 30 頁)

(21)出願番号 特願2001-145592(P2001-145592)
(22)出願日 平成13年5月15日(2001.5.15)
(31)優先権主張番号 特願2000-141492(P2000-141492)
(32)優先日 平成12年5月15日(2000.5.15)
(33)優先権主要国 日本 (JP)
(31)優先権主張番号 特願2000-141493(P2000-141493)
(32)優先日 平成12年5月15日(2000.5.15)
(33)優先権主要国 日本 (JP)
(31)優先権主張番号 特許2000-398746(P2000-398746)
(32)優先日 平成12年12月27日(2000.12.27)
(33)優先権主要国 日本 (JP)

(71)出願人 000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地
(72)発明者 佐見 雄一
大阪府門真市大字門真1006番地 案下電器
産業 株式会社内
(72)発明者 畠山 喬
大阪府門真市大字門真1006番地 案下電器
産業 株式会社内
(74)代理人 100000446
弁理士 中島 司朗

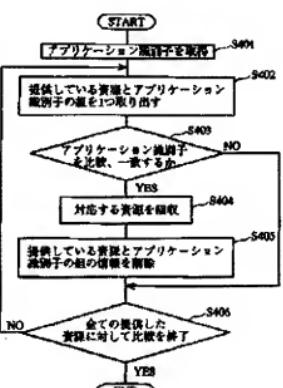
最終頁に続く

(54)【発明の名稱】 アプリケーション実行装置及び方法

(57)【要約】

【課題】 資源回収をカーネルによらずに実行するアプリケーション実行装置を提供すること、Javaミドルウェアを再起動することなく、連続的にJavaアプリケーションを実行することができるJavaアプリケーション実行装置を提供すること、及びガベージコレクション処理の負荷を軽減したJavaアプリケーション実行装置を提供すること。

【解決手段】 ライブドリブルはアプリケーションからの資源の性別の要求に応じて、資源を提供したときに、資源の提供を要求したアプリケーションを識別する識別子を取得し、提供した資源と取得したアプリケーション識別子を組にしてテーブルに保持する。そして、アプリケーションが終了すると、終了したアプリケーションのアプリケーション識別子を取得し、取得した識別子に対応する資源をテーブルから特定し、対応した資源を解放する。



(2)

特開2002-259146

2

【特許請求の範囲】

【請求項1】 アプリケーションに資源を提供する少なくとも1つ以上のライブラリ部とカーネル部を備えるアプリケーション実行装置であって、
前記カーネル部は、

アプリケーションが終了すると、終了したアプリケーションがどれであるかを資源を提供した各ライブラリ部に通知する通知手段を有し、
各ライブラリ部は、

通知手段から通知されると、終了したアプリケーションに提供した資源を回収する回収手段を有することを特徴とするアプリケーション実行装置。

【請求項2】 前記回収手段は、

アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源保持手段とを有することを特徴とする請求項1記載のアプリケーション実行装置。

【請求項3】 前記各ライブラリ部はさらに、

アプリケーションからの要求に応じて資源を提供する提供手段と、

資源を提供したアプリケーションがどれであるかの通知を前記カーネル部から受け取り、前記テーブルに、資源を提供したアプリケーションと提供した資源を対応付けて記録して登録する登録手段と、

登録した資源が前記回収手段により回収されると、回収された資源と対応するアプリケーションとの対応関係を示す組を前記テーブルから削除する削除手段とを有することを特徴とする請求項2記載のアプリケーション実行装置。

【請求項4】 前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、

前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部にアプリケーションが終了すれば前記通知手段を行うよう依頼する依頼手段とを有することを特徴とする請求項3記載のアプリケーション実行装置。

【請求項5】 前記依頼手段は、コールバック間数の呼び出しを前記カーネル部に依頼し、
前記カーネル部は、アプリケーションが終了すると、依頼されたコールバック間数を呼び出して実行することにより、前記通知手段として前記通知手段を行うことを特徴とする請求項4記載のアプリケーション実行装置。

【請求項6】 前記通知手段は、アプリケーション毎に前記通知手段を行う複数の通知部を有し、
前記各ライブラリ部はさらに、

アプリケーションからの要求に応じて資源を提供する提

前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部に当該アプリケーションが終了すれば前記通知手段を行うよう当該アプリケーションに対応する通知部に依頼する依頼手段とを有することを特徴とする請求項6記載のアプリケーション実行装置。

【請求項7】 前記依頼手段は、前記各ライブラリ部が前記通知手段を受け取るために生成した資源回収インスタンスのメソッドの呼び出しにより前記通知手段を行うよう前記通知手段に依頼し、
10 依頼された前記通知部は、前記当該アプリケーションが終了すると、資源回収インスタンスのメソッドを呼び出すことにより、前記通知手段を行うことを特徴とする請求項6記載のアプリケーション実行装置。

【請求項8】 前記回収手段は、
アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有し、
前記通知手段は、複数のアプリケーションが同時に終了した場合に、終了した複数のアプリケーションがどれであるかを、資源を提供した各ライブラリ部に通知し、
前記資源特定手段は、通知された複数のアプリケーションに応する複数の資源を前記テーブル保持手段により持続し、
前記回収手段は、特定された複数の資源を回収することを特徴とする請求項7記載のアプリケーション実行装置。

【請求項9】 前記各ライブラリ部は、前記資源として、チャーナー、NFCデコーダ、リモコン、ファイリシステム、メモリ、モデムの何れかをアプリケーションに提供することを特徴とする請求項8記載のアプリケーション実行装置。

【請求項10】 前記回収手段は、
アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することを特徴とする請求項9記載のアプリケーション実行装置。

【請求項11】 前記各ライブラリ部は、前記資源として、チャーナー、NFCデコーダ、リモコン、ファイリシステム、メモリ、モデムの何れかをアプリケーションに提供することを特徴とする請求項10記載のアプリケーション実行装置。

【請求項12】 前記回収手段は、
アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することを特徴とする請求項11記載のアプリケーション実行装置。

【請求項13】 前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することを特徴とする請求項12記載のアプリケーション実行装置。

【請求項14】 前記各ライブラリ部は、前記資源として、チャーナー、NFCデコーダ、リモコン、ファイリシステム、メモリ、モデムの何れかをアプリケーションに提供することを特徴とする請求項13記載のアプリケーション実行装置。

【請求項15】 前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することを特徴とする請求項14記載のアプリケーション実行装置。

【請求項16】 前記各ライブラリ部はさらに、
資源を提供したアプリケーションがどれであるかの通知を前記カーネル部から受け取り、前記テーブルに、資源を提供したアプリケーションと提供した資源を対応付けて登録する登録手段と、
前記カーネル部は、アプリケーションが終了すると、依頼されたコールバック間数を呼び出して実行することにより、前記通知手段として前記通知手段を行うことを特徴とする請求項15記載のアプリケーション実行装置。

【請求項17】 前記各ライブラリ部はさらに、
アプリケーションからの要求に応じて資源を提供する提供手段と、
前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することを特徴とする請求項16記載のアプリケーション実行装置。

(3)

特開2002-259146

4

3

された資源と対応するアプリケーションを前記テーブルから削除する削除手段とを有することを特徴とする請求項1の記載のアプリケーション実行装置。

【請求項12】 前記各ライブラリ部はさらに、アプリケーションからの要求に応じて資源を提供する機会手段と、前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部にアプリケーションが終了すれば前記通知を行うよう依頼する依頼手段とを有することを特徴とする請求項9記載のアプリケーション実行装置。

【請求項13】 前記依頼手段は、コードバック関数の呼び出しを前記カーネル部に依頼し、

前記カーネル部は、アプリケーションが終了すると、依頼されたコードバック関数を呼び出して実行することにより、前記通知手段として前記通知を行うことを特徴とする請求項12記載のアプリケーション実行装置。

【請求項14】 前記通知手段は、アプリケーション毎に前記通知を行う複数の通知部を備え、

各ライブラリ部はさらに、アプリケーションからの要求に応じて資源を提供する機会手段と、

前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部に当該アプリケーションが終了すれば前記通知を行うよう当該アプリケーションに付随する通知部に依頼する依頼手段とを有することを特徴とする請求項9記載のアプリケーション実行装置。

【請求項15】 前記依頼手段は、前記各ライブラリ部が前記通知を受け取るために生成した資源回収インスタンスのメソッドの呼び出しにより前記通知を行うよう前記通知部に依頼し、

依頼された前記通知部は、前記当該アプリケーションが終了すると、資源回収インスタンスのメソッドを呼び出すことにより、前記通知を行うことを特徴とする請求項14記載のアプリケーション実行装置。

【請求項16】 前記依頼手段は、

アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、

前記通知手段によって通知されたアプリケーションに機会手段を前記テーブル保持手段により特定する資源判定手段とを有し、

前記通知手段は、複数のアプリケーションが同時に終了した場合に、終了した複数のアプリケーションがどれであるかを、資源を提供したライブラリ部に通知し、

前記資源判定手段は、通知された複数のアプリケーションに対応する複数の資源を前記テーブル保持手段により特定し、

前記回収手段は、特定された複数の資源を回収すること

置。

【請求項17】 前記通知手段は、さらにアプリケーションが終了した時と中断した時の何れかの時に、前記各ライブラリ部に通知し、

前記各ライブラリ部はさらに、通知されたアプリケーションが終了したか、中断したかに応じて、当該アプリケーションに提供した資源を回収するか否かを判定する判定手段を有し、

前記回収手段は、さらに資源を回収すると判定された場合に、通知されたアプリケーションに提供した資源を回収することを特徴とする請求項9記載のアプリケーション実行装置。

【請求項18】 前記通知手段は、アプリケーション毎に前記通知を行う複数の通知部を備え、

前記各ライブラリ部はさらに、アプリケーションからの要求に応じて資源を提供する機会手段と、

前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部に当該アプリケーションが終了した時と中断した時の何れかの時に、前記通知部に依頼する依頼手段とを有することを特徴とする請求項12記載のアプリケーション実行装置。

【請求項19】 前記依頼手段は、前記各ライブラリ部が前記通知を受け取るために生成した資源回収インスタンスのメソッドの呼び出しにより前記通知を行うよう前記通知部に依頼し、

依頼された前記通知部は、前記当該アプリケーションが終了した時と中断した時の何れかの時に、前記資源回収インスタンスのメソッドを呼び出すことにより、前記通知手段に依頼し、

前記判定手段は、前記通知を受け取ると、アプリケーションが終了したか、中断したかについての情報を取得し、前記情報に応じて当該アプリケーションに提供した資源を回収するか否かを判定することを特徴とする請求項14記載のアプリケーション実行装置。

【請求項20】 アプリケーションに資源を提供するOS部とJavaミドルウェア部を備えるアプリケーション実行装置であって、

前記Javaミドルウェア部は、アプリケーションとアプリケーションに応じるタスクとタスクを構成するスレッドの対応関係を示すテーブルを保持する第1テーブル保持手段と、

アプリケーション終了の指示を受けると、前記第1テーブル保持手段を参照して終了するアプリケーションに応じるタスクとタスクを構成するスレッドの対応関係を示すテーブルを有し、

前記OS部は、アプリケーション毎にアプリケーションを実行するタス

(4)

特開2002-259146

5

前記タスクを構成するアプリケーション用スレッドを復数生成するスレッド生成手段と、

前記アプリケーション用スレッドを実行することにより、アプリケーションのプログラムコードを実行し、アプリケーションが要求する資源を提供し、提供した資源と前記スレッドが属するタスクとの対応関係を示すテーブルを保有する管理手段と、
前記通知手段から通知されたタスクに対応する資源を管理手段より特定し、アプリケーションに提供されていた資源を回収する回収手段とを有することを特徴とするアプリケーション実行装置。

【請求項21】 前記Javaミドルウェア部はさらに、デバイスの状態の変化を、実行中のアプリケーションに通知してくれるよう依頼する伝達手段と、
前記依頼に応じて、前記変化を検知すると、前記アプリケーションに前記変化があったことの通知をする状態変化通知手段とを有することを特徴とする請求項20記載のアプリケーション実行装置。

【請求項22】 前記依頼手段は、前記変化があったことの通知を待ち受けるリスナーへの呼び出しを前記状態変化通知手段に依頼し、
前記状態変化通知手段は、前記変化を検知すると、依頼された前記リスナーへの呼び出しを行ことにより、前記通知手段することを特徴とする請求項21記載のアプリケーション実行装置。

【請求項23】 前記状態変化通知手段は、前記リスナーへの呼び出しを実行する専用スレッドを生成し、当該専用スレッドを実行することにより前記リスナーへの呼び出しを行い、

前記Javaミドルウェア部はさらに、前記リスナーと前記専用スレッドと前記アプリケーションとの対応関係を示すテーブルを保持する第2テーブル保持手段と、

前記依頼手段が前記リスナーへの呼び出しを前記状態変化通知手段に依頼する毎に、前記テーブルを参照し、前記リスナーと対応関係にあるアプリケーションが前記テーブルに保持されているか否かを判定し、保持されている場合は、当該アプリケーションに対応づいて当該リスナーへ前記テーブルに追加するテーブル更新手段と、
前記状態変化通知手段は、前記テーブル更新手段によつて当該アプリケーションが保持されていると判定された場合は、前記専用スレッドを新たに生成せず、保持されないと判定された場合は、前記専用スレッドを新たに生成することを特徴とする請求項22記載のアプリケーション実行装置。

【請求項24】 前記専用スレッドは、スレッド間の情報伝達を行うキューに、前記変化が発生したことと示す情報を保持されているか否かを監視し、情報が保持されている場合に対応する前記リスナーへの呼び出しを行うことを特徴とする請求項23記載のアプリケーション実行

6

【請求項25】 前記専用スレッドは、スレッド間の情報伝達を行うキューに、前記変化が発生したことと示す情報を保持される前はクエイド状態であり、キューに情報を削除されると実行状態に移行し、対応する前記リスナーの呼び出しを行ふことを特徴とする請求項23記載のアプリケーション実行装置。

【請求項26】 前記Javaミドルウェア部はさらに、前記タスクを構成する前記アプリケーション用スレッドとは別に、前記Javaミドルウェア部に必要な資源を確保する資源確保用スレッドを生成する資源確保用スレッド生成手段と、

生成した資源確保用スレッドを実行することにより前記Javaミドルウェア部に必要な資源を確保する資源確保手段とを有し、

前記回収手段は、前記資源確保手段によって確保された前記Javaミドルウェア部に必要な資源を回収することなく、前記通知手段から通知されたタスクに対応する資源を前記管理手段より特定し、アプリケーションに提供されていた資源を回収することを特徴とする請求項20記載のアプリケーション実行装置。

【請求項27】 アプリケーションに資源を提供するOS部とミドルウェア部を構えるアプリケーション実行装置であつて、
前記ミドルウェア部は、

前記OS部からの要求に応じて資源の提供を要求したアプリケーションがどれであるかを通知し、アプリケーションが終了すると、終了したアプリケーションがどれであるかを通知する通知手段を有し、
前記OS部は、

資源の提供を要求したアプリケーションがどれであるかの通知を要求したアプリケーションがどれであるかの通知を要求する要求手段と、
通知されたアプリケーションと資源との対応関係を示すテーブルを保持する資源管理テーブル手段と、
通知手段から終了したアプリケーションがどれであるかが通知されると、前記資源管理テーブル手段により終了したアプリケーションに対応する資源を特定し、特定した資源を回収する資源回収手段とを有することを特徴とするアプリケーション実行装置。

【請求項28】 前記通知手段は、

49 資源の提供を要求したアプリケーションをロードしたクラスローダを特定するローダ特定手段と、
アプリケーションをロードしたクラスローダとアプリケーションとの対応関係を示すテーブルを保持するテーブル手段と、
前記ローダ特定手段によって特定されたクラスローダに對応するアプリケーションを前記テーブルから特定するアプリケーション特定手段とを有することを特徴とする請求項27記載のアプリケーション実行装置。

【請求項29】 前記ローダ特定手段は、アプリケーション

(5)

特開2002-259148

8

ックを参照することにより、資源の提供を要求したアプリケーションをロードしたクラスロードを特定することを特徴とする請求項2-8記載のアプリケーション実行装置。

【請求項3-0】 前記0部はさらに、アプリケーション毎にアプリケーションを特定するIDを割当てる割当て手段を有し、

前記通知手段は、前記割当て手段によって割当てられたIDを通知することにより、資源を提供したアプリケーションがどれであるかを通知し、終了したアプリケーションがどれであるかを通知することを特徴とする請求項2-9記載のアプリケーション実行装置。

【請求項3-1】 前記0部はさらに、

アプリケーション毎にアプリケーションを特定するIDを割当てる割当て手段を有し、

前記通知手段は、前記割当て手段によって割当てられたIDを通知することにより、資源を提供したアプリケーションがどれであるかを通知し、終了したアプリケーションがどれであるかを通知することを特徴とする請求項2-8記載のアプリケーション実行装置。

【請求項3-2】 前記0部はさらに、アプリケーション毎にアプリケーションを特定するIDを割当てる割当て手段を有し、

前記通知手段は、前記割当て手段によって割当てられたIDを通知することにより、資源を提供したアプリケーションがどれであるかを通知し、終了したアプリケーションがどれであるかを通知することを特徴とする請求項2-8記載のアプリケーション実行装置。

【請求項3-3】 アプリケーションに資源を提供する複数のライブラリ部とカーネル部を備えるアプリケーション実行装置に用いられるログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記プログラムは、請求項1、17の何れか1項に記載の手段をコンピュータに実現させることを特徴とする記録媒体。

【請求項3-4】 アプリケーションに資源を提供する05部とミドルウェア部を備えるアプリケーション実行装置に用いられるログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記プログラムは、請求項2-0に記載の手段をコンピュータに実現させることを特徴とする記録媒体。

【請求項3-5】 アプリケーションに資源を提供する05部とミドルウェア部を備えるアプリケーション実行装置に用いられるログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記プログラムは、請求項2-7に記載の手段をコンピュータに実現させることを特徴とする記録媒体。

【請求項3-6】 ガベージコレクションが必要なアプリケーション用のメモリヒープ領域を監査するアプリケーション

アプリケーションが起動される毎に、メモリヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域獲得手段と、

起動されたアプリケーションに、前記分割ヒープ領域獲得手段によって獲得された分割ヒープ領域を割当てる割当手段と、

アプリケーションが終了する毎に、当該アプリケーションに割当てられた分割ヒープ領域を解放するメモリ解放手段とを備えることを特徴とするアプリケーション実行装置。

【請求項3-7】 前記アプリケーション実行装置はさらに、

アプリケーションに関連するオブジェクトに対して、当該アプリケーションに割当てられた分割ヒープ領域内でオブジェクト領域を獲得するオブジェクト領域獲得手段と、

前記分割ヒープ領域を対象にガベージコレクションを行うガベージコレクション手段とを備えることを特徴とする請求項3-6記載のアプリケーション実行装置。

【請求項3-8】 前記アプリケーション実行装置はさらに、前記ガベージコレクション手段によって対象となる分割ヒープ領域のガベージコレクションが行われている間、当該分割ヒープ領域を使用するアプリケーションの実行を停止させ、他の分割ヒープ領域を使用するアプリケーションは継続して実行させるロック手段を備えることを特徴とする請求項3-7記載のアプリケーション実行装置。

【請求項3-9】 前記アプリケーション実行装置はさらに、起動中のアプリケーションと分割ヒープ領域との対応関係表示テーブルを保持するテーブル手段を備え、

前記通知手段は、終了したアプリケーションに対応する分割ヒープ領域を前記7テーブル手段から特定して解放することを特徴とする請求項3-6記載のアプリケーション実行装置。

【請求項4-0】 前記アプリケーション実行装置はさらに、起動中のアプリケーションと分割ヒープ領域との対応関係表示テーブルを保持するテーブル手段を備え、

前記通知手段は、さらに、アプリケーションが起動される毎に、当該アプリケーションと分割領域との新たな対応関係をテーブルに追加することを特徴とする請求項3-6記載のアプリケーション実行装置。

【請求項4-1】 ガベージコレクションが必要なアプリケーション用のメモリヒープ領域を管理するアプリケーション実行装置であって、

メモリヒープ領域全体をシステムヒープ領域として割当てるシステムヒープ領域割当手段と、

システム関連のオブジェクト領域をシステムヒープ領域内に獲得するオブジェクト領域獲得手段と、

アプリケーションが起動される毎に、システムヒープ

(6)

特開2002-259146

9

19

枝と。

起動されたアプリケーションに、獲得された分割ヒープ領域を割当てる割当手段と、
アプリケーションの終了時に、当該アプリケーションに
対して割当られた分割ヒープ領域を解放するメモリ解放手段とを備えることを特徴とするアプリケーション実行装置。

【請求項4-2】 前記アプリケーション実行装置はさらに、
システムヒープ領域内部を対象に、分割ヒープ領域
を単位にガベージコレクションを行うガベージコレクション手段を備えることを特徴とする請求項4-1記載のア
プリケーション実行装置。

【請求項4-3】 ガベージコレクションが必要なアプリ
ケーション用のメモリヒープ領域を管理するメモリヒー
プ管理方法であって、

アプリケーションが起動される前に、メモリヒープ領域
から分割ヒープ領域を獲得する分割ヒープ領域獲得ステ
ップと、
起動されたアプリケーションに、前記分割ヒープ領域獲
得ステップによって獲得された分割ヒープ領域を割当て
る割当ステップと、
アプリケーションが終了する前に、当該アプリケーシ
ョンに割当された分割ヒープ領域を解放するメモリ解放
ステップとを含むことを特徴とするメモリヒープ管理方
法。

【請求項4-4】 前記メモリヒープ管理方法はさらに、
アプリケーションに関連するオブジェクトに対して、当
該アプリケーションに割当された分割ヒープ領域内
でオブジェクト領域を獲得するオブジェクト領域獲得ステ
ップと、

オブジェクト領域獲得ステップにおいてオブジェクト域
域に失敗したとき、該々の分割ヒープ領域を対象に
ガベージコレクションを行うガベージコレクションス
テップとを含むことを特徴とする請求項4-3記載のメモ
リヒープ管理方法。

【請求項4-5】 前記メモリヒープ管理方法はさらに、
前記ガベージコレクションステップによって対象となる
分割ヒープ領域のガベージコレクションが行われている
間、当該分割ヒープ領域を使用するアプリケーションの
み実行を作させ、他の分割ヒープ領域を使用するア
プリケーションは継続して実行させるロックステップを含
むことを特徴とする請求項4-4記載のメモリヒープ管理
方法。

【請求項4-6】 前記メモリヒープ管理方法はさらに、
起動中のアプリケーションと分割ヒープ領域との対応関
係示すテーブルを保持するテーブル保持ステップを含
み。

前記メモリ解放ステップは、終了したアプリケーション
に対応する分割ヒープ領域を前記テーブルから特定して

プロセス方法。

【請求項4-7】 前記メモリヒープ管理方法はさらに、
起動中のアプリケーションと分割ヒープ領域との対応関
係示すテーブルを保持するテーブル保持ステップを含
み。

前記割り当てステップは、さらに、アプリケーションが
起動される毎に、当該アプリケーションと分割領域との
新たな対応関係を前記テーブルに追加することを特徴と
する請求項4-3記載のメモリヒープ管理方法。

10 【請求項4-8】 ガベージコレクションが必要なア
プリケーション用のメモリヒープ領域を監視するメモリヒー
プ監視方法であって、

メモリヒープ領域全体をシステムヒープ領域として割当
てるシステムヒープ領域割当ステップと、
システム開始のオブジェクト領域をシステムヒープ領域
内に獲得するオブジェクト領域獲得ステップと、

アプリケーションが起動される毎に、システムヒープ領
域から分割ヒープ領域を獲得する分割ヒープ領域獲得ス
テップと、

20 起動されたアプリケーションに、獲得された分割ヒー
プ領域を割り当てる割当ステップと、

アプリケーションの終了時に、当該アプリケーションに
対して割当された分割ヒープ領域を解放するメモリ解放
ステップとを含むことを特徴とするメモリヒープ監視
方法。

【請求項4-9】 前記メモリヒープ監視方法はさらに、
分割ヒープ領域獲得ステップが分割ヒープ領域の獲得に
失敗したとき、システムヒープ領域内部を対象に、分
割ヒープ領域を単位にガベージコレクションを行なうガベ
ージコレクションステップを含むことを特徴とする請求項
4-8記載のメモリヒープ監視方法。

【請求項5-0】 ガベージコレクションが必要なア
プリケーションを実行する装置に導入されるプログラムを
記録したコンピュータ読み取り可能な記録媒体であ
って、

前記プログラムは、請求項4-3～4-9の何れか1項に記
載のステップをコンピュータに実行させることを特徴と
する記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、アプリケーション
実行装置において、アプリケーション終了により不必要
になった資源の回収を実行する技術に関する。

【0002】

【従来の技術】 一般的なアプリケーション実行装置にお
いては、オペレーティングシステムのカーネルが、ア
プリケーション終了等により不要になった資源（メモリ、
キーボードなどの各種デバイス）の回収を行っている。
ここで、「回収」とは、カーネル等のオペレーティングシ

ーションから取り返すことをいう。カーネルは資源の回収を行うために、資源の回収の手順をデバイス毎に記憶している。これにより、アプリケーションが使用していたデバイスが不要になると、カーネルは当該デバイスの回収の手順に従って、資源を回収することができ、資源を効率よく利用することができる。

【0003】また、Javaアプリケーション実行装置において、例えばPCでは、メモリ以外の他の資源の回収については、Javaミドルウェア(いわゆるパーキャラマシン)自身は資源の回収を行わず、Javaミドルウェアが終了した時に、カーネルによって資源の回収が行われる。一方、メモリの回収は、Javaミドルウェアによる「ガベージコレクション」により行われる。「ガベージコレクション」とは、メモリ領域の不要になった領域を回収することをいう。Javaでは、アプリケーションのロードは、アプリケーションを構成する複数のクラスの各インスタンス(以下オブジェクトと記述)をメモリ中にロードすることにより行われる。メモリ領域は、実行中のアプリケーションのためにロードされるオブジェクトで共用され、メモリ領域には、ロードされる際に領域が割り当てられる。従って、複数のアプリケーションから連続的にオブジェクトがメモリ領域にロードされた場合には、1つのアプリケーションからロードされた各オブジェクトがメモリ領域に不適な形で割り当てられることになる。メモリ領域がロードされたオブジェクトで不連続に占められると、メモリ不足を解消するため、ガベージコレクションが行われる。

【0004】

【発明が解決しようとする課題】しかしながら、上記に説明した資源回収処理の技術には以下に示す問題点がある。第1に、アプリケーション実行装置におけるカーネルによる資源の回収処理の場合には、新規にデバイスをアプリケーション実行装置に追加する毎に、追加されたデバイスの回収手順を新たにカーネルに記録せなければならぬという問題がある。

【0005】第2に、Javaアプリケーション実行装置における資源の回収処理の場合には、Javaミドルウェアを終了することなく、アプリケーションを直進的に実行していくと、カーネルによるメモリ以外の資源の回収は、Javaミドルウェアが終了するまで行われないため、1つのアプリケーションが終了しても、そのアプリケーションに提供された資源は回収されないので、次のアプリケーションが同じ資源を必要とする場合には、次のアプリケーションが実行できなくなるという問題がある。

【0006】第3に、不連続な形でオブジェクトに割り当てられたメモリ領域が不要となった場合、メモリ領域のガベージコレクションの負荷が大きくなるという問題がある。上記の問題点に鑑み、本発明は、資源回収をカーネルによらずに実行するアプリケーション実行装置を提

出続的にJavaアプリケーションを実行することができるJavaアプリケーション実行装置を提供すること、及びガベージコレクション処理の負荷を軽減したJavaアプリケーション実行装置を提供することを目的とする。

【0007】

【課題を解決するための手段】上記課題を解決するために、本発明に係るアプリケーション実行装置は、アプリケーションに資源を提供する少なくとも1つ以上のライブラリ部とカーネル部を備えるアプリケーション実行装置であって、前記カーネル部は、アプリケーションが終了すると、終了したアプリケーションがどれであるかを資源を提供した各ライブラリ部に通知手段を有し、各ライブラリ部は、通知手段から通知されると、終了したアプリケーションに提供した資源を回収する回収手段を有することを特徴とする。本アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することもできる。

【0008】また、本発明に係るアプリケーション実行装置は、アプリケーションに資源を提供する部とJavaミドルウェア部を備えるアプリケーション実行装置であって、前記Javaミドルウェア部は、アプリケーションとアプリケーションに付随するタスクとタスクを構成するスレッドの対応関係を示すテーブルを保持する第1テーブル保持手段と、アプリケーション終了の指示を受けると、前記第1テーブル保持手段を参照して終了するアプリケーションに対応するタスクがどれであるかを通知する通知手段とを有し、前記2部は、アプリケーション毎にアプリケーションを実行するタスクを生成するタスク生成手段と、前記タスクを構成するアプリケーション用スレッドを複数生成するスレッド生成手段と、前記アプリケーション用スレッドを実行することにより、アプリケーションのプログラムコードを実行し、アプリケーションが要求する資源を提供し、提供した資源と前記スレッドが割り当てるタスクとの対応関係を示すテーブルを保持する管理手段と、前記通知手段から通知されたタスクに対応する資源を管理手段より持り、アプリケーションに提供されていた資源を回収する回収手段とを有することを特徴とする。本アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することもできる。

【0009】本発明に係るアプリケーション実行装置は、ガベージコレクションが必要なアプリケーションを実行する装置であって、アプリケーションが起動される毎に、メモリヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域獲得手段と、起動されたアプリケーションに、前記分割ヒープ領域獲得手段によって獲得された分割ヒープ領域を割り当てる手段と、アプリケーシ

(3)

特開2002-259146

14

れた分割ヒープ領域を解放するメモリ解放手段とを備えることを特徴とする。本アプリケーション実行装置における各手段等をステップとするメモリヒープ管理方法としたり、それらステップをコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することもできる。

【00101】

【発明の実施の形態】(実施の形態1) 本実施の形態における各アプリケーション実行装置においては、アプリケーションが終了等する毎にカーネル部ではなく、ライブラリ部が各アプリケーションに提供する資源の回収を行う。ここで、「アプリケーション」は、プログラムとプログラムが実行されることによる機能の回数を指すものとする。「資源」とは、チャネル、MPEGデータオデコーダ、MPEGビデオオデコーダ、キーボード入力デバイス、リモコン入力デバイス、マウス入力デバイス、トラックボール入力デバイス、ミューティクス、セマフォア、ネットワークデバイス、シリアルデバイス、IEEE1394(シリアルインターフェース)、USB(Universal Serial Bus)デバイスインターフェース、ファイリシステム、ディスクライバ、メモリ、モモ等をいう。ライブラリ部は、アプリケーションの要求に応じて資源を制御(資源の確保、登録、回収)するいわゆるデバイスドライバである。「回収」とは、カーネル等のオペレーションシステム側がアプリケーションに提供した資源をアプリケーションから取り戻すことをいう。

【0011】具体的には、ライブラリ部はアプリケーションからの資源の提供の要求に応じて、資源を提供したときに、資源の提供を要求したアプリケーションを識別する識別子を取得し、提供した資源と取扱ったアプリケーション識別子を組にしてテーブルに格納する。そして、アプリケーションが終了すると、終了したアプリケーションのアプリケーション識別子を取得し、取扱した識別子に対する資源をテーブルから特定し、特定した資源を回収する。

【0012】以下本アプリケーション実行装置の実施の形態について、図面について説明する。図1aは、アプリケーション実行装置1の構成を示すブロック図である。アプリケーション実行装置1は、アプリケーション入力部11、アプリケーション記憶部12、カーネル部13、ライブラリ格納部14、デバイス部15、指示過剰部16から構成される。

【0013】アプリケーション入力部11は、フロッピーディスクドライブ、CDドライブ、ネットワークインターフェースボード、放送受信機等で構成され、実行すべきアプリケーションを受け付け、アプリケーション記憶部12に記録させる。アプリケーション記憶部12は、ROM、ROM、ハードディスク、CDドライブ、フロッピーディスク等で構成され、アプリケーショ

内蔵されているアプリケーションを記憶する。

【0014】カーネル部13は、逐次実行部13a、資源回収通知部13b、識別子提供部13cから構成され、アプリケーション記憶部12が記憶するアプリケーションを実行する。逐次実行部13aは、実行するアプリケーションがバイナリープログラムの場合、バイナリープログラムをそのまま実行するCPU等で構成される。実行するアプリケーションがJavaバイトコード(Javac)、米国のSun Microsystems, Inc.の商標である。のような中間コードの場合は中間コードを逐次解析し、実行するモジュール(パーザルマシンやインタプリタ)等で構成される。逐次実行部13aは、アプリケーション記憶部12に記憶するアプリケーションを読み出し、実行する。さらに、実行するアプリケーションからライブラリ格納部14に格納されるライブラリ部が資源の提供を要求し、ライブラリ部からアプリケーションが要求する資源の提供を受ける。

【0015】資源回収通知部13bは、アプリケーションが終了等する毎に当該アプリケーションに資源を提供

20しているライブラリ部に対しても、資源回収の指示と当該アプリケーション識別子を通知する。具体的には、資源回収通知部13bは、ライブラリ格納部14が格納するライブラリ部から、コールバック関数の置き受けつけ、指定通知部16からアプリケーションの終了又は中断の通知を受けると、置きされているコールバック関数を呼び出して実行し、当該アプリケーションの識別子をライブラリ部に通知する。

【0016】ここで、「コールバック関数の置き受けつけ」とは、ライブラリ部がアプリケーションの要求に応じて、当該アプリケーションに既存した資源の提供を開始したときに、「当該ライブラリ部に、当該アプリケーションが終了又は中断すれば中断又は終了したアプリケーションがどれであるかを、コールバック関数を呼び出して実行することにより通知してくれるよう」資源回収通知部13bに依頼することをいう。

【0017】図2は、コールバック関数を実装したC言語プログラムの一例を示す。CALLBACK.hは、コールバック関数の型名を示す。このコールバック関数は返り値がなく、引数に整数の値をとる。図3は、コールバック関数の型名と解除と実行を記述したC言語プログラムの例を示す。行番号6-14は要約のプログラム例を示し、行番号16-26は具体的なプログラム例を示し、行番号22-26は実行のプログラム例を示す。

【0018】識別子提供部13cは、ライブラリ部からの要求に応じて、アプリケーションの識別子を提供する。ライブラリ格納部14は、資源があるデバイスを制御する複数のライブラリ部14a~14nからなる。各ライブラリ部は、デバイス部15の対応するデバイスを制御する(例えばライブラリ部14aは、デバイス部15a

(9)

特開2002-259146

16

る。)。実際にはライブラリ格納部4は、ROMやハードディスクで構成され、各ライブラリ部では、デバイス制御用の制御群を記述したプログラムがアプリケーションに呼び出されて実行されることによりデバイスの制御が実現されている。

【0019】具体的には、各ライブラリ部は、以下に示す処理を行うことにより資源を制御する。アプリケーションの要求に応じて資源を提供し、資源回収通知部13bにコールバック函数を登録する。次にアプリケーション識別部13cから資源を要求したアプリケーションのアプリケーション識別子を取得し、提供した資源の資源名と組にしてテーブルに保持する。さらに、登録したコールバック函数が資源解放通知部13bにより実行され、回収すべき資源を使用しているアプリケーションのアプリケーション識別子が通知されると、保持しているテーブルから通知されたアプリケーション識別子に対応する資源名を特定し、特定した資源名の資源を回収する。

【0020】次に各ライブラリ部がアプリケーションの要求に応じて行う資源提供の処理について図4を参照して説明する。図4は、各ライブラリ部が行う資源提供の処理を示すフローチャートである。各ライブラリ部は、アプリケーションの要求に応じて、資源を確保し(S301)、コールバック函数を既に登録しているか否かを判断(例えばフラグを設定し、登録・未登録を判断)し

(S302)、登録していない場合(S302:N)は、資源回収通知部13bにコールバック函数を登録する(S303)。

【0021】次に識別子供供部13cを呼び出し、資源を要求したアプリケーションのアプリケーション識別子を取得し(S304)、取得したアプリケーション識別子と確保した資源の資源名を組にしてテーブルに保持し(S305)。資源をアプリケーションに提供する(S306)。登録している場合(S302:Y)は、コールバック函数を新たに登録することなく、S304～S306の処理を実行する。

【0022】上記テーブルの構造図5に示す。図5は、ファイルシステムを制御するライブラリ部(以下ライブラリ部と呼ぶ。)が保持するアプリケーション識別子と資源名(ファイル名)とを組にしてテーブルを示す。図5-(1)は、ライブラリ部が資源解放通知部13bにコールバック函数を登録し、アプリケーション識別子「1」のアプリケーションに対し、ファイル名「a.txt」の資源を提供していることを示す。アプリケーションがライブラリ部の函数を呼び出すことにより、資源の提供を要求すると、ライブラリ部は、ファイル名「b.txt」の資源を確保し、識別子供供部13cからアプリケーション識別子「2」を取得する。この場合、ライブラリ部のコールバック函数は、既に登録済みなので再登録はされない。ライブラリ部は、取得したアプリケーション識別子「2」と確保した資源のファイル名

に示すようにテーブルを更新する。

【0023】次に各ライブラリ部が行う資源の回収処理について図6を参照して説明する。図6は、各ライブラリ部が行う資源の回収処理を示すフローチャートである。コールバック函数が呼び出されると、各ライブラリ部はアプリケーション識別子と資源回収通知部13dより取扱し(S401)、テーブルに保持するアプリケーション識別子と資源名の組を1つ取り出し(S402)、取り出したアプリケーション識別子と資源回収通知部13dより取扱したアプリケーション識別子を比較する(S403)。アプリケーション識別子が一致する場合は、取り出した組の資源名を回収し(S404)、テーブルから当該組を解除し(S405)。テーブルに保持する全ての組との比較が終了している場合(S405:Y)は処理を終了する。例えば図5-(2)のテーブルの例で、取得したアプリケーション識別子が「2」の場合には、テーブルからアプリケーション識別子「2」と対応するファイル名「b.txt」の組が削除される(図5-(3))。全ての組との比較が終了していない場合(S405:N)は次の組を取り出し(S402)、同様の処理を繰り返す(S403～S405)。アプリケーション識別子が一致しない場合は、テーブルに保持する全ての組との比較が終了している場合(S405:Y)は処理を終了する。全ての組との比較が終了していない場合(S405:N)は次の組を取り出し(S402)。同様の処理を繰り返す(S403～S405)。

【0024】デバイス15～15aは、モモリー、ディスプレイ、入力デバイス(キーボード、マウス等)、ファイルシステム、ネットワークデバイスなどを含み、各デバイスはライブラリ格納部4に格納する各ライブラリ部を介してなされるアプリケーションからの制御に応じて作動する。指示通知部16は、アプリケーションの開始、停止、再開、終了の処理をする。

【0025】アプリケーションの開始の処理とは、逐次実行部13aにアプリケーション記憶部12が記憶するアプリケーションの実行を指示することをいう。アプリケーション停止の処理とは、逐次実行部13aが実行しているアプリケーションの終了を指示し、資源回収通知部13bにアプリケーションの終了を通知することをいう。

【0026】アプリケーション再開の処理とは、逐次実行部13aに中断しているアプリケーションの実行を指示することをいう。アプリケーション終了の処理とは、逐次実行部13aが実行しているアプリケーションの終了を指示し、資源回収通知部13bにアプリケーションの終了を通知することをいう。

【0027】本実施形態においては、アプリケーション終了又は中断時に資源をアプリケーションに提供した各ライブラリ部に通知されるアプリケーション識別子は、1つであったが、複数のアプリケーションが同時に終了

(10)

特開2002-259146

18

プラリ部に通知することとしてよい。この場合の各ライブラリ部が行う資源の回収処理のフローチャートを図7に示す。各ライブラリ部が回収する資源を提供した複数のアプリケーション識別子を資源回収通知部13bより取得し(S751)、取得したアプリケーション識別子を1つ取り出し(S752)、テーブルに保持するアプリケーション識別子と資源名の組を1つ取り出し(S753)、取り出したアプリケーション識別子と資源名として保持し(S754)。アプリケーション識別子が一致する場合(S754:Y)は、対応する資源名を回収対象の資源名として保持し(S755)。テーブルから当該組を削除し(S756)、テーブルに保持する全ての組との比較が終了している場合(S757:Y)は、次のアプリケーション識別子を取得し(S758、S752)、同様の処理を繰り返す。S757で取得した全てのアプリケーション識別子に対する比較が終了すると(S758:Y)、S755で回収対象の資源名として保持している資源名に対応する資源を一括して回収する(S759)。

【0028】これにより、ライブラリ部は、効率的に資源回収を行うことができる。例えば、「(1つの)ライブラリ部が管理する資源が、複数のアプリケーションの終了によって全て回収される場合、當接テーブルを初期化する処理だけで迅速に資源を回収することができる。

(実施形態2) 簡略形態1においては、ライブラリ部は、アプリケーションが終了した場合に呼び出されるコールバック関数をライブラリ部が登録したが、本実施形態においては、実施形態1のコールバック関数に相当する資源回収インスタンスを生成し、アプリケーションが終了する毎にアプリケーションに提供した資源を回収する。ここで、「資源回収インスタンス」は、ライブラリ部が構築するクラス(*JNIEnv*バイトコードのプログラム)から生成され、各種の情報とその情報を取り出すための機能であるメソッドが記述されたプログラムであり、資源回収インスタンスは、資源の構造を要求したアドリケーションと提供した資源の情報とこれららの情報を取り出したためのメソッドを構成している。このメソッドをプログラム上で呼び出すことにより保持している情報を取り出される。

【0029】以下本アドリケーション実行装置の実施の形態について、図面を用いて説明する。図8は、アドリケーション実行装置20の構成を示すブロック図である。アドリケーション実行装置20は、アドリケーション入力部11、アドリケーション記憶部12、03部23、ライブラリ接続部24、デバイス部15、指示通知部16から構成される。アドリケーション入力部11、アドリケーション記憶部12、デバイス部15、指示通知部16は、実施形態1の実行装置10に含まれるものと同一であるので、説明を省略する。

アドリケーション管理部23bから構成され、アドリケーション記憶部12が記憶するJavaアドリケーション(以下アドリケーションといふ。)を実行する、バーチャルマシン部23aは、アドリケーション記憶部12からアドリケーションをロードし、アドリケーションのバイトコードを逐次解釈し、アドリケーションを実行する。また、ライブラリ接続部24に格納される各ライブラリ部を介してデバイス部15に格納される各デバイスを制御する。実際には、バーチャルマシン部23aが実行するアドリケーションからデバイス制御を記述したプログラムである各ライブラリ部のクラスが呼び出され、クラスのプログラムが実行されることにより、各デバイスの制御が実現される。

【0031】アドリケーション管理部23bは、実行されているアドリケーションを実行し、また、実施形態1の識別子提供部13cに相当する機能を有する。具体的には、アドリケーション記憶部23bは、アドリケーション情報を格納する処理を実行するクラスから、処理を実行するインスタンスをアドリケーション毎に生成し、生成した各インスタンス(ここでは、アドリケーション情報インスタンスと呼ぶ。)に、実行されている各アドリケーションのアドリケーション識別子を格納させ、生成した各アドリケーション情報インスタンス(図8の23c~23e)を用いて実行されているアドリケーションを監視する。また各ライブラリ部がアドリケーションから資源の提供を要求されたとき、要求したアドリケーションを待機する識別子としてアドリケーション情報インスタンスを提供する。ここで、各アドリケーションの識別子を各アドリケーション情報インスタンスに格納される代わりに、アドリケーション情報インスタンスそのものをアドリケーションの識別子としてもよい。

【0032】各アドリケーション情報インスタンスは、実施形態1の資源回収通知部13bに相当する機能を有する。具体的には、以下の処理を行う。各アドリケーション情報インスタンスは、各ライブラリ部から資源回収インスタンスの登録を受け付ける。ここで、「資源回収インスタンスの登録」とは、ライブラリ部がアドリケーションに新たな資源を提供したときに、「当該ライブラリ部に、当該アドリケーションが終了又は中断すれば、当該アドリケーションがどれであるかを通知してくれるよう」に当該アドリケーション識別子を格納することをいう。上記の通知は、登録時に生成される資源回収インスタンスに実装されるメソッドを呼び出すことにより実行される。資源回収インスタンスは各ライブラリ部が格納する各クラスから、以下に示すように生成される。各クラスは、「インターフェース」を実装することにより、「インターフェース」が定義するメソッドを実装する。「インターフェース」は、クラスに実装されるべきメソッド

(11)

特開2002-259146

19

20

る。この例では、ResourceCollectionListenerインターフェースがupdateメソッドを定義している。各ライブラリ部が格納する基クラスは、このインターフェースを実装することにより、updateメソッドに資源回収処理を記述したクラスを用意し、用意したクラスから当該インターフェースを実装した資源回収インスタンスが生成され、アプリケーション情報インスタンスに生成した資源回収インスタンスが登録される。この処理により、生成された資源回収インスタンスには、インターフェースが定義するメソッド（ここでは、updateメソッド）が実装されていることが保証され、上記の通りはこのインターフェースが定義するメソッドを呼び出すことにより実行される。

【0033】また、アプリケーションが終了又は中断した場合に、当該アプリケーション識別子を格納するアプリケーション情報インスタンスは、登録された資源回収インスタンスに実装されているメソッド（ここでは、updateメソッド）を呼び出すことにより、当該アプリケーションに資源を提供しているライブラリ部に上記の通知をする。

【0034】さらに、アプリケーション情報インスタンスは、登録された資源回収インスタンスを削除するメソッドを実現してもよい。アプリケーションの状態を知るためにメソッドを格納してもよい。図10は、上記のメソッドを格納するアプリケーション情報インスタンスを生成するクラスの一例を示す。図10において、アプリケーション情報インスタンスを生成するクラスであるapplicationPropsは、アプリケーション名(applicationName)に相当する内部変数と、アプリケーションの状態を知るためにgetStatuseメソッド、資源回収インスタンス(resourceCollectionListener)を登録するaddListenerメソッド、資源回収インスタンスの登録を削除するremoveListenerメソッドを定義している。

【0035】ライブラリ格納部2-4は、複数のライブラリ部2-4～2-4からなる。各ライブラリ部は、デバイス部1-5のデバイス1-5～1-5を削除し、アプリケーションからの要求に応じて資源あるデバイス提供の処理と、提供した資源回収の処理を行う。実際にはライブラリ格納部2-4は、ROMやハードディスクで構成され、各ライブラリ部では、デバイス制御用のプログラムであるライセンス群に記述されたプログラムがアプリケーションに呼び出されて実行されることによりデバイスの制御が実現されている。

【0036】最初に資源提供の処理について説明する。各ライブラリ部は、アプリケーションから資源の提供を要求されると、アプリケーションに要求された資源を提供し、アプリケーションが當該部2-3を呼び出し、当該アプリケーションに対応するアプリケーション情報インスタンスを取得し、資源回収クラスから資源回収イン

アプリケーション情報インスタンスに登録する。さらに、各ライブラリ部は、提供した資源の資源名と当該アプリケーション情報インスタンスとを組にして資源回収インスタンスに保持する。

【0037】図11に資源回収クラスの一例を示す。図11においては、ファイルシステムを調査する資源回収クラスが登載されている。具体的には、資源回収インスタンス生成時に作成される「フィールド」と呼ばれる内部変数app（アプリケーション情報インスタンスに相当する）とname（資源のファイル名に相当する）の情報とアプリケーション終了時又は定期的に呼び出され、資源回収の処理を実行するupdateメソッドが定義されている。図12は、図11の資源回収クラスが生成した資源回収インスタンスの例を示す。9001は資源回収インスタンスを保持するインスタンステーブルを示し、902、903は、生成された資源回収インスタンスを示す。図12-(1)は、資源回収情報インスタンス902が提供した資源名（ここでは、ファイル名「a.txt」）とアプリケーション情報インスタンス（ここで「1」とする。）を組にして保持していることを示す。図12-(2)は、ライブラリが図12-(1)とは別のアプリケーション情報インスタンス「2」に登録されたアプリケーション情報（ファイル名「b.txt」のファイル）を提供し、新たに資源回収インスタンス903を生成したことを示す。

【0038】以下図13を用いて、資源提供の処理をさらに詳しく説明する。図13は、各ライブラリ部が行う資源回収の処理を示すフローチャートである。各ライブラリ部は、アプリケーションから資源の提供を要求されると、要求に応じて資源を確保し（S1001）、アプリケーション管理部2-3から当該アプリケーションに応対するアプリケーション情報インスタンスを取得する（S1002）。次に保持している資源回収インスタンスの1つを取り出し（S1003）。アプリケーション管理部2-3から取り出したアプリケーション情報インスタンスと取り出された資源回収インスタンスに保持されるアプリケーション情報インスタンスを比較する（S1004）。アプリケーション情報インスタンスが一致する場合（S1004:Y）は、当該アプリケーション情報インスタンスを保持する資源回収インスタンスに確保した資源の資源名を保持し（S1005）。確保した資源を当該アプリケーションに提供する（S1009）。一方、S1004において、アプリケーション情報インスタンスが一致しない場合（S1004:N）には、保持している他の全ての資源回収インスタンスに対してアプリケーション情報インスタンスの比較を行い（S1006）。一致するものがなければ（S1006:Y）、新たに資源回収インスタンスを生成し、生成した資源回収インスタンスに、アプリケーション管理部2-3から取得したアプリケーション情報インスタンスと確保した資源

(12)

特開2002-259146

21.

22

収インスタンスを資源の確保を要求したアプリケーションに対するアプリケーション情報インスタンスに登録し(S1008)、確保した資源をアプリケーションに提供する(S1009)。

【0039】次に資源回収の処理について説明する。ライブラリ部は、アプリケーション終了又は中断時に当該アプリケーション識別子を登録するアプリケーション情報インスタンスから対応する資源回収インスタンスのメソッド呼び出されると、当該資源回収インスタンスに保持されている資源名に対応する資源を回収する。上記の場合において、アプリケーション情報インスタンスにアプリケーションの状態を知るためのメソッド(以下アプリケーション状態メソッドといふ)と資源回収インスタンスの登録を削除するメソッドとを用いた場合の資源回収の処理について説明する。図14は、アプリケーション情報インスタンスがアプリケーション状態メソッドを有している場合のライブラリ部が行う回収の処理を示すフローチャートである。ライブラリ部は、アプリケーション情報インスタンスから登録した資源回収インスタンスのメソッドが呼び出されると、資源回収インスタンスに保持されているアプリケーション情報インスタンスのアプリケーション状態メソッドを呼び出し、アプリケーションが中断されたか、終了されたかの情報を取得し(S1010)、取得した情報を基して資源を回収するか否かを判定し(S1012)、資源を回収する場合は、当該資源回収インスタンスに保持されている資源名に対応する資源を回収し(S1013)、当該資源回収インスタンスの登録を削除するメソッドを呼び出し、当該メソッドを実行することにより、当該資源回収インスタンスを解散する(S1014)。

【0040】なお、S102における判定は、アプリケーションが使用している全ての資源ではなく、特定の一部の資源を回収するという判定であってもよい。この場合、S102における資源回収インスタンスの解散は、全ての資源が回収されたときに行われる。

(実施の形態3) 実実施の形態におけるアプリケーション実行装置は、アプリケーション毎にスレッドを生成し、生成したスレッドでアプリケーションが要求する資源を確保し、アプリケーションが終了するまで終了したアプリケーションに提供した資源の解放を、生成したスレッドと対応付けられたタスク単位でOC部が行うことにより、不要な資源を回収し、Javaミドルウェア(いわゆるバーチャルマシン)を終了することなく次のアプリケーションの実行を可能とするものである。ここで、「スレッド」とは、アプリケーションのプログラムコードを逐次実行するコンテキスト(一連の処理の流れ)である。Javaで記述されたアプリケーションは、複数のスレッドを生成し、それぞれのスレッドがアプリケーションのプログラムコードを並行して実行できることができる

ションを実行する複数のスレッドの集合である。本実施形態においてアプリケーションに相当される資源は、この「タスク」と間違つて管理される。

【0041】以下、本アプリケーション実行装置の実施の形態について四面を用いて詳細に説明する。図15は、アプリケーション実行装置30の構成を示すブロック図である。アプリケーション実行装置30は、アプリケーション入力部11、アプリケーション記憶部12、Javaミドルウェア部33、OC部34、ハードウェア部35、指示通知部36で構成される。アプリケーション入力部11、アプリケーション記憶部12は、実施形態1の実行装置10に含まれるものと同一であるので、説明を省略する。

【0042】Javaミドルウェア部33は、VM(バーチャルマシン)部33a、アプリケーション管理部33b、クラスライブラリ格納部33cから構成され、アプリケーション記憶部12が記憶するアプリケーションを実行する。VM部33aは、アプリケーションを実行する。具体的には、VM部33aは、以下に示す処理を行う。バイト

コードで記述されたアプリケーションをハードウェア部35で記述されるCPU35aが実行できるバイナリコードに逐次翻訳し、翻訳したバイナリコードをCPU35aが実行させる。

【0043】また、VM部33aは、自己の動作とアプリケーションの実行に必要な資源を確保する。具体的には、VM部33aは、各クラスライブラリ格納部33cに含まれる各クラスライブラリ部を呼び出し、各クラスライブラリ部が介して自己の動作に必要な資源を確保する。また、アプリケーション実行中にアプリケーションから資源の確保を要求されると、各クラスライブラリ格納部33cに含まれる各クラスライブラリ部を呼び出し、各クラスライブラリ部が介してアプリケーションが要求した資源を確保する。

【0044】また、VM部33aは、アプリケーションから処理終了したスレッドの削除を要求されると、カーキル34aに依頼してスレッドを削除させる。アプリケーション管理部33bは、アプリケーション起動の処理、アプリケーション実行中に追加又は削除されるスレッドの管理、アプリケーション終了の処理を行う。

【0045】最初に、アプリケーション起動の処理について説明する。アプリケーション管理部33bは、指示通知部36からアプリケーションの開始指示を受けると、カーキル34aにタスクと、タスクに属する最初のスレッドを生成させ、VM部33aで生成した最初のスレッドで、アプリケーションを実行することにより、VM部33aの動作に必要な資源を確保させ、アプリケーションを起動させる。

【0046】次にアプリケーション実行中に追加又は削除されるスレッドの管理について説明する。アプリケ

(13)

特開2002-259146

24

ションに対応して生成されたタスクとタスクを構成するスレッドとの対応関係を示すテーブルを保持し、アプリケーションの要求によりスレッドが追加又は削除される毎にテーブルを更新することにより、スレッドをアプリケーションとタスクに対応づけて管理する。図16にテーブルの例を示す。図16-(1)では2つのアプリケーションが実行中であり、アプリケーションID「1」のアプリケーションに対し、タスクID「201」のタスクが生成され、タスクID「201」のタスクはスレッドID「1」と「2」のスレッドで構成され、アプリケーションID「2」のアプリケーションに対し、タスクID「202」のタスクが生成され、タスクID「202」のタスクはスレッドID「4」と「5」、「6」のスレッドにより構成されていることを示し、図16-(2)は、アプリケーションID「1」のアプリケーションを実行中に、新たなスレッド(スレッドID「7」のスレッド)が生成され、生成されたスレッドのスレッドID「7」がテーブルに追加されたことを示し、図16-(3)は、アプリケーション「2」のアプリケーションを実行中にスレッドID「5」のスレッドが削除されたことを示す。

[0047]最後に、アプリケーション終了の処理について説明する。図17は、アプリケーション終了時にアプリケーション管理部3-3bが行う処理を示すフローチャートである。アプリケーション管理部3-3bは、指示通知部3-6からアプリケーション終了の指示を受けると、カーキル3-4aが終了したアプリケーションに対応するタスクのIDを通知し、資源回収を実施し(S801)、次に、クラスライブラリ格納部3-3cに含まれる各クラスライブラリ部に終了するアプリケーションのIDを通知し(S802)、最後に終了したアプリケーションに対応するタスクを生成するために確保したメモリを解放してタスクを終了させる(S803)ことになり、アプリケーション終了の処理を行う。例えば、図16-(3)の例では、アプリケーションID「2」のアプリケーションが終了した場合、アプリケーション管理部3-3bは、カーキル3-4aがアプリケーションID「2」に対応するタスクID「202」を通知し、カーキル3-4aがタスクID「202」に対応する資源の回収を実施し、次にクラスライブラリ部に終了したアプリケーションID「2」を通知し、最後にメモリを解放して生成したタスク「202」及びその中に含まれるスレッド「4」と「6」を終了させる。

[0048]クラスライブラリ格納部3-3cは、複数のクラスライブラリ部3-3c1~3-3c3からなり、各クラスライブラリ部は、ハードウェア部5に含まれる資源であるデバイスを制御し、資源提供の処理、リスナーの登録とリスナーの呼び出しの処理を行う。実際には、クラスライブラリ格納部3-3cは、ROMやhardtディスクなどで構成され、各クラスライブラリ部では、デバイス制

ムがアプリケーションに呼び出されて実行されることによりデバイスの制御が実現されている。

[0049]最初に、各クラスライブラリ部が行う資源検出の処理について説明する。クラスライブラリ部は、W部3-3aが実行するアプリケーションから呼び出され、資源あるデバイスの検出を要求されると、W部3-4のライブラリ格納部3-4に指名される当該デバイスに対応するライブラリ部を呼び出し、要求された資源を検出する。

[0050]次に各クラスライブラリ部が行うリスナー登録の処理について説明する。クラスライブラリ部は、W部3-3aが実行するアプリケーションから呼び出され、リスナー登録され、登録されたリスナーを呼び出す処理をするスレッド(以後専用スレッドと呼ぶ。)を生成する。クラスライブラリ部による専用スレッドの生成はアプリケーションを実行しているスレッドで実行される。クラスライブラリ部は、アプリケーションを実行しているスレッドに属するスレッドとして、リスナーを呼び出す専用スレッドを生成する。

[0051]ここで、リスナーとはイベント(ユーザーによってリモコン入力操作がされたことやモードムが相手先から切断されたことなどによるデバイスの状態変化)が発生したことの連絡を待ち受けけるクラスまたはインターフェースであり、リスナーには、イベントがあった時に呼び出されるメソッドが定義され、当該メソッドを呼び出し、メソッドを実行することによりイベントの処理が行われる。登録されたリスナーは、イベント発生時に呼び出される。例えば、ユーザーによるリモコン操作入力等のイベントがあったときに登録されたリスナーに定義されたメソッドが呼び出され、呼び出されたメソッドを実行することによってイベントが処理される。また、「リスナー登録」とは、アプリケーションが「イベント」が発生すれば、当該アプリケーションに通知してくれるよう)当該イベントを監視するクラスライブラリ部に登録することをいう。上記の通知は、当該クラスライブラリ部がリスナー登録時に当該リスナーに定義されるメソッドを呼び出すことにより実現される。

[0052]また、クラスライブラリ部はリスナー登録される際、アプリケーション管理部3-3bを呼び出し、リスナー登録をするアプリケーションのIDを取得する。

アプリケーション管理部3-3bは、以下の処理を行ふことによりアプリケーションのIDを判定し、クラスライブラリ部に判定したIDを提供する。アプリケーション管理部3-3bは呼び出しを行ったスレッドのスレッドクラスのインスタンスを取得し、当該インスタンスからインスタンスが保持するハッシュコード値を取り出すメソッドであるhashcode関数を用いて、呼び出しを行ったインスタンスのID(スレッドID)を判定し、アプリケーション管理部3-3bが管理するアプリケーションIDとスレッド

(14)

特開2002-259146

25

26

るアプリケーションIDを特定し、クラスライブラリ部に特定したIDを提供する。ここで、取り出されるハッシュコード値は、「ID部3_3」が効率よく、クラスから生成されたインスタンスを管理できるように、各インスタンスに割り振った識別番号（ID）である。各クラスライブラリ部は、アプリケーション管理部3_3bから提供されたアプリケーションIDと生成した専用スレッドのIDと登録されたリスナーのIDとに対応付けてテーブルに保持する。

【0053】図18は、クラスライブラリ部が行うリスナー登録の処理を示すフローチャートである。クラスライブラリ部は、ID部3_3aが実行するアプリケーションからリスナー登録のために呼び出されると、アプリケーションIDを取得し（S501）、登録されたリスナー 専用スレッド、アプリケーションIDとの対応関係を示すテーブルから、アプリケーションIDと専用スレッドとリスナーの組の1つを取り出し（S502）、当該アプリケーションIDと取得したアプリケーションIDを比較する（S503）。

【0054】IDが一致する場合（S503:Y）は、当該ID部に対応付けて新たにリスナー登録されたリスナーをテーブルに登録する（S504）。IDが一致しない場合（S503:N）は、テーブルに保持されている他の組を取り出し（S505～S502）、同様の処理を繰り返す。IDが一致する場合が多く、ID比較を終了すると（S505:Y）、新たに登録されたリスナーに対応する専用スレッドを生成し（S506）、取得したアプリケーションIDと生成した専用スレッドと新たに登録されたリスナーの組をテーブルに追加し、保持する（S507）。

【0055】これにより、テーブルに保持されたアプリケーションが複数回リスナー登録をする場合に、アプリケーションIDに対応する専用スレッドをテーブルから特定することができ、アプリケーションによって登録された複数のリスナーを、特定した1つの専用スレッドに対応させてテーブルにまとめて保持することができる。従って、1つの専用スレッドから対応する複数のリスナーを呼び出すことができる。アプリケーションからリスナー登録される毎に専用のスレッドを生成する必要がなく、生成するスレッドの数を少なくて済むことができる。

【0056】図19は、クラスライブラリ部が保持するテーブルの構成である。図19-(1)は、アプリケーションID「1」のアプリケーションがリスナー「L1」とリスナー「L2」を登録し、これらのリスナーに対応する専用スレッド「thread10」が生成されていることを示す。図19-(2)は、図19-(1)のテーブルにリスナー「L6」が登録されて、テーブルが更新されたことを示す。具体的には、リスナー「L6」がリスナー登録されたと、クラスライブラリ部は、アプリケーション管理部3

のアプリケーションIDを取得し（ここでは、ID「2」を取得したものとする）。図19-(1)のテーブルに保持されるアプリケーションID「1」と一致するか否かを判定し、一致しないと判定すると、新規に専用スレッドを生成し、生成した専用スレッド「thread20」とアプリケーションID「2」とリスナー「L6」とを対応させてテーブルに保持する。図19-(3)は、アプリケーションID「1」のアプリケーションが終了し、テーブルに保持されていたアプリケーションID「1」と対応する専用スレッド「thread10」、リスナー「L1」とリスナー「L4」がテーブルから削除されたことを示す。

【0057】次にイベントが発生してから、クラスライブラリ部が専用スレッドから登録されたリスナーを呼び出すまでに行われる処理について説明する。この処理は、2つの処理から構成される。最初に行われる処理は、イベントが発生したことと連動する処理であり、この処理は、ID部3_4により生成されたスレッドにより実行される。図20は、イベントが発生したことと通知する処理を示すフローチャートである。クラスライブラリ部は、ID部3_4のライブラリ基盤部3_4bに含まれるライブラリ部からイベントがあったことを通知されると、アプリケーションIDと専用スレッドと登録されたリスナーとの組を保持したテーブルから組の1つを取り出し（S601）、取り出した組の専用スレッドのキューにリスナー呼び出し必要な呼び出し情報を保持する（S602）。ここで、キューは、スレッド間の情報伝達を行ない、「情報を1つ保持する」、「情報を1つ取り出す」という2つの操作をすることができる。同様の処理をテーブルに保持する全ての組について行う（S603～S60

3)。

【0058】次に行われる処理は、登録されたリスナーを呼び出す処理であり、この処理は、クラスライブラリ部が生成した専用スレッドにより実行される。図21は、クラスライブラリ部が行うリスナー呼び出しの処理を示すフローチャートである。リスナー呼び出し部は、専用スレッドを実行させることにより、キューに呼び出し情報が保持されているか否かを監視し（S701）、キューに呼び出し情報があれば（S701:Y）、それを取り出し（S702）、テーブルから専用スレッドに対応するリスナーを検索し（S703）、リスナーを呼び出す（S704）。専用スレッドに対応するリスナーが複数ある場合は、このリスナー呼び出し処理（S703、S704）を対応する全てのリスナーについて行う（S703～S705）。図22は、キューを用いて専用スレッドからリスナーを呼び出す手順の例を模式化した図である。

【0059】図22において、411、412はそれぞれ、図19の専用スレッド「thread10」と専用スレッド「thread20」に対して用意されたキューを示す。また、401、402はそれぞれ、専用スレッド「thread10」

(15)

特開2002-259146

27

28

のライブラリ協調部3-4 b/c協調されるライブラリ部からイベントが発生したことを通知する0部3-4により生成されたスレッドであり、破線4-21、4-22は情報を保持されることを示し、黒丸4-31、4-32は保持された情報を示し、破線4-41、4-42は保持された情報を取り出すことを示す。

【0060】 クラスライブラリ部は、ライブラリ部からイベントが発生したことをスレッド4-03により通知されると、リスナー呼び出しに必要な情報(4-31、4-32)をキュ-411、4-12に保持する(4-21、4-22)。専用スレッド4-01、4-02は、キュ-41
1-4-12に情報を保持しているか否かを監視し、情報が削除されていると、その情報を取り出し対応するリスナーを呼び出す。すなわち、専用スレッド4-01はリスナー「[1]」とリスナー「[4]」を、専用スレッド4-02はリスナー「[6]」を呼び出す。

【0061】 0部3-4は、カーネル3-4aとライブラリ協調部3-4bから構成され、Javaミドルウェア部3-3を作動させることにより、アプリケーションを間接的に動作させる。カーネル3-4aは、アプリケーションに提供された資源をアプリケーションに対するタスク単位で管理する。具体的には、カーネル3-4aは、アプリケーション管理部3-3b/cからの指示により、アプリケーションに付与するタスクとタスクを履歴する最初のスレッドを生成し、最初のスレッドで0部3-3の動作に必要な資源を確保する。またアプリケーションが要求するスレッドを生成し、生成したスレッドでアプリケーションが要求する資源を提供し、提供した資源の資源名をスレッドが属するタスクIDに対応付けで保持する。アプリケーション管理部3-3aから終了したアプリケーションに付与するタスクIDの通知を受けると、通知されたタスクIDに対応づけて保持された資源名の資源を全て回収する。

【0062】 ライブラリ協調部3-4aは、資源であるデバイスを制御する複数のライブラリ部3-4b1～3-4b4からなる。各ライブラリ部は、ハードウェア部3-5の対応するデバイスを制御する(例えばライブラリ部3-4b1では、デバイス3-5b1を、ライブラリ部3-4b2では、デバイス3-5b2を制御する。)。ライブラリ協調部3-4bは、ROMやハードディスクで構成され、各ライブラリ部では、デバイス制御用の間数群を記述したプログラムがクラスライブラリのクラスに呼び出されて実行されることによりデバイスの制御が処理されている。

【0063】 ハードウェア部3-5は、0部3-4やJavaミドルウェア部3-3を作動させるCPU3-5a及びデバイス3-5b1～3-5b4で構成される。CPU3-5aは、カーネル3-4aが生成したスレッドを実行することによりアプリケーションのプログラムコードを実行する。デバイス3-5b1～3-5b4は、メモリーデバイス、ディスプレイデバイス、入力デバイス(キーボード、マウス等)、フ

デバイスはクラスライブラリ及びライブラリ部を介してなされるアプリケーションからの制御に応じて作動する。

【0064】 指示通知部3-6は、アプリケーションの開始及び終了をアプリケーション管理部3-3bに指示する。具体的には、アプリケーション管理部3-3bにアプリケーション開始の指示をし、外部からアプリケーション終了の指示を受けると、アプリケーション管理部3-3bにアプリケーションの終了を指示する。以上、本発明の実施の形態について説明したが、本発明はこの実施の形態に限界されないのは言うまでもない。

【0065】 なお、図2-1のフローチャートでは、クラスライブラリは専用スレッドを実行させて、常にキュ-1に呼び出し情報が保持されているか否かを監視する(S7-01)としたが、S7-01において、専用スレッドをウエイト状態にして、キュ-1に呼び出し情報が保持されたとき、専用スレッドのウエイト状態を解除し、専用スレッドを実行させて、S7-02～S7-03の処理を行わせることとしてもよい。これにより、資源を効率的に利用することができる。

【0066】 なお、図1-7のS801とS802は順序を逆にしてもよい。また、アプリケーション毎に生成されるスレッドの数は図1-6の例に限らない。この例より多くても少なくともよい。また、本実施の形態において、資源の管理は、カーネル3-4aが行うとしたが、ライブラリ協調部3-4bに含まれる各ライブラリ部が行うこととしてもよい。この場合、資源供給の際にカーネル3-4aが資源を提供するタスクを問い合わせ、カーネル3-4aより当該タスクのIDの通知を受けると、通知を受けたタスクIDに供給する資源の資源名を組にして保持し、カーネル3-4aより特定のタスクIDの資源回収の指示があると、指示されたタスクIDに對応する資源名を保持した後から特定し、特定した資源名の資源を回収する。また、本実施の形態においては、W部3-3aは、アプリケーションを実行するタスクに含まれる最初のスレッドでW部3-3aに必要な資源が確保されることが、アプリケーションを実行するスレッドから独立したシステム専用スレッドをクラスライブラリ部で生成させ、生成させたシステム専用スレッドで必要な資源を確保することとしてもよい。この場合のクラスライブラリ部の行う処理を図2-3を用いて説明する。図2-3は、W部3-3aがシステム専用のスレッドを獲得する処理を示すフローチャートである。クラスライブラリ部は、アプリケーションを実行するスレッドで、実行すべき処理(例えばアプリケーションが要求する資源を提供)し(S5401)、次にシステム専用のスレッドを生成し(S5402)、生成したシステム専用スレッドで実行すべき処理(例えばW部3-3aが必要な資源を提供する処理)を実行し(S5403)、処理が終了すると処理結果をアプリケーションに通知する(S5404)

(16)

特開2002-259146

39

専用のスレッドの処理は、任意の回数、交互に繰り返し実行してもよいし、並行して実行してもよい。また、55
40の通知は、不要としてもよい。

【0067】これにより、VM部3.3.3が必要とする資源は、システム専用のスレッドで確保されるので、VM部3.3.3はアプリケーションが終了しても自己に必要な資源を再確保することなく、次のアプリケーションを実行することができる。また、上記のシステム専用スレッドでクラスライブラリ部が自己に必要な資源を確保することとしてもよい。例えば、図19に例として示すアプリケーションID0とスレッドインスタンスとリソースインスタンスの対応関係を示すテーブルを保持するのに必要なメモリを上記システム専用スレッドで確保することとしてよい。

【0068】また、本実施の形態においては、クラスライブラリ部が専用のスレッドを生成するとしたが、アプリケーション実行中に画面に描画するための機能を提供するComponentクラスからComponentインスタンスを生成する時に生成したインスタンスに格納されるpaintメソッドを呼び出すため呼び出し用スレッドを生成することとしてもよい。さらに、アプリケーションによるリソース登録のためのクラスライブラリ部による専用スレッドの生成をアプリケーション起動時に完了することとしてもよい。

【0069】これにより、クラスライブラリ部はリソース登録する際に専用スレッドの有無を確認する必要がなくなり、リソース登録の処理を短時間で完了することができる。

(実施の形態4) 本実施の形態におけるアプリケーション実行装置においては、各ライブラリ部がアプリケーションに提供する資源の回収を行い、アプリケーションが終了する際にアプリケーションが使用した資源を回収する。

【0070】具体的には、以下の処理を行う。各ライブラリ部は各クラスライブラリ部を介してアプリケーションから資源の種類を要求されると、要求された資源を提供し、各クラスライブラリ部に資源を要求したアプリケーションのIDを取得し、取得したアプリケーションIDと提供した資源の資源名とを組にしてテーブルに保持する。ライブラリ部は、カーネル4.4より終了したアプリケーションIDの通知を受けると、通知されたアプリケーションIDに対応する資源名を、保持しているテーブルから特定し、特定した資源名の資源を全て回収する。

【0071】以下本アプリケーション実行装置の実施の形態について、画面を用いて説明する。図24は、アプリケーション実行装置4.0の構成を示すブロック図である。アプリケーション実行装置4.0は、アプリケーション入力部1.1、アプリケーション記憶部1.2、Javaミドルウェア部4.3、OS部4.4、ハードウェア部4.5、指示

【0072】アプリケーション入力部1.1、アプリケーション記憶部1.2は、実施形態1の実行装置1.0に含まれるものと同一であるので、説明を省略する。Javaミドルウェア部4.3は、VM部4.3.a、アプリケーション管理部4.3.b、クラスライブラリ格納部4.3.cから構成され、アプリケーション記憶部4.2が記憶するアプリケーションを実行する。

【0073】VM部4.3はアプリケーション記憶部1.2からアプリケーションをロードし、アプリケーションのバイトコードを逐次解析し、アプリケーションを実行する。アプリケーション管理部4.3.bは、VM部4.3.cが実行するアプリケーションの起動の処理、実行中のアプリケーションの監視、アプリケーション終了の処理を行う。

【0074】はじめにアプリケーション起動の処理について説明する。アプリケーション管理部4.3.bは、クラスローダーのインスタンスをアプリケーション毎に生成し、クラスローダーのインスタンスを用いてアプリケーションを構成する全てのクラスファイルをロードする。ここで、クラスローダーは、Javaプログラムで使用されるクラスファイルをメモリに読み込むクラスである。

【0075】次に実行中のアプリケーションの監視について説明する。アプリケーション管理部4.3.bは、起動したアプリケーションを構成するクラスファイルをロードするクラスローダーのインスタンスと当該アプリケーションに割当てられたIDを組み込んだテーブルを保持する。なお、IDはOS部4.4のカーネル4.4によって、起動されたアプリケーション毎にそれぞれ異なるIDが割当てられる。図25は、アプリケーション管理部4.3.bが保持するクラスローダーのインスタンスと対応するアプリケーションIDを組みしたテーブルの例を示す。

【0076】次にアプリケーション終了の処理について説明する。アプリケーション管理部4.3.bは、制御部4.6からアプリケーション終了の指示を受けると、カーネル4.4.aとクラスライブラリ格納部4.3.cが協調するクラスライブラリ部に終了するアプリケーションIDを通知する。アプリケーション管理部4.3.bは、さらに資源をアプリケーションに提供したクラスライブラリ部から呼び出され、当該アプリケーションのアプリケーションID通知の要求があると、以下の処理を行って当該アプリケーションIDを特定し、クラスライブラリに通知する。

【0077】アプリケーション管理部4.3.bは、クラスの呼び出し情報を記録したスタッカを参照し、アプリケーションをロードしたクラスローダーのインスタンスを取得し、保持しているテーブルから取得したクラスローダーのインスタンスに対応するアプリケーションIDを特定し、特定したアプリケーションIDを当該クラスライブラリ部に通知する。ここで、スタッカは、クラスのメソッドの作成順序として割当てられるメモリ領域であり、メソッドの呼び出しによりクラスの呼び出しが行われる毎

てられ、スタック中に呼び出し情報を保持するクラスが構成に採納される。呼び出し情報には、メソッドの呼び出し元を特定する情報が格納される。

【0078】以下具体例を挙げて、アプリケーションIDを特定する処理について詳しく説明する。図26はアプリケーション管理部4 3bが生成したクラスローダのインスタンスとスタックとの関係を示す模式図である。ここでは、クラスローダのインスタンス6 30 1がアプリケーションを構成するクラスAのメソッドを呼び出し、呼び出されたクラスAがアプリケーションを構成する別のクラスのメソッドを呼び出し、次にクラスBのアプリケーションを構成する別のクラスCを呼び出し、最後にクラスDがクラスライブラリ部に格納されるクラスEのメソッドを呼び出し、資源の確保を要求する場合の例を用いて説明する。この場合、スタック中には、呼び出しがされる毎に、呼び出されたクラスの資源領域が割り当てられ、割り当てられた資源領域に呼び出し情報が格納される。6 30 1は、アプリケーション管理部4 3bが管理するクラスローダの格納場所を表し、ここでは、2つのクラスローダ6 30 1と6 30 2が格納されている。6 31 0は、スタックを表し、6 31 1～6 31 4はスタック6 31 0に格納されているクラスA～Eが保持された呼び出し情報を示す。アプリケーション管理部4 3bは、クラスBC呼び出されたクラスライブラリ部からアプリケーションIDの特定要求があると、スタック6 31 0に格納された呼び出し情報6 31 4～6 31 1を順に参照することにより、アプリケーションをロードしたクラスローダのインスタンス6 30 1を取得する。アプリケーション管理部4 3bは、保持しているテーブルを参照し、取得したクラスローダのインスタンスに対応するアプリケーションIDを特定する。

【0079】クラスライブラリ格納部4 3cは、複数のクラスライブラリ部4 3c1～4 3c3からなり、各クラスライブラリ部は、ライブラリ格納部4 4bに含まれる対応する各ライブラリ部を介して、ハードウェア部4 5に含まれる資源である各デバイスを制御する。実際にには、クラスライブラリ格納部4 3cは、ROM等で構成され、各クラスライブラリ部では、各デバイス制御用のプログラムであるクラス群に記述されたプログラムがアプリケーションに呼び出されて実行されることにより各デバイスの制御が実現されている。

【0080】各クラスライブラリ部は、アプリケーションから呼び出され、資源であるデバイスの提供を要求されると、ライブラリ格納部4 4bに含まれる当該デバイスに対応する各ライブラリ部を呼び出し、要求された資源を当該アプリケーションに提供する。さらに、クラスライブラリ部は当該アプリケーションのアプリケーションIDをアプリケーション管理部4 4bから取得し、当該ライブラリ部に取得したアプリケーションIDを通知する。

部4 3bからアプリケーション終了の通知と終了したアプリケーションのアプリケーションIDを通知を受け、各ライブラリ部に受け取ったアプリケーションIDを通知し、資源の回收を指示する。ライブラリ格納部4 4bは、デバイスを制御する複数のライブラリ部4 4b1～4 4b3からなる。各ライブラリ部は、ハードウェア部4 5の制御するデバイスを制御する（例えばライブラリ部4 4b1は、デバイス4 5b1を、ライブラリ部4 4b2は、デバイス4 5b2を制御する。）。

【0082】実際にはライブラリ格納部4 4bは、ROMやハードディスクで構成され、各ライブラリ部では、デバイス制御用の間断群を記述したプログラムが各クラスライブラリ部のクラスに呼び出されて実行されることにより実現してあるデバイスの制御が実現されている。具体的には、各ライブラリ部はクラスライブラリ部からの呼び出しに応じて、アプリケーションIDに対して資源を提供し、クラスライブラリ部から当該アプリケーションのアプリケーションIDの通知を受け、提供した資源の資源名と受け取ったアプリケーションIDを組にしてテーブルに保持する。カーネル4 4aより終了したアプリケーションのアプリケーションIDの通知を受けると、保持しているテーブルから通知を受けたアプリケーションIDに応じる資源を特定し、特定した資源の資源を回收する。図27は、提供した資源の資源名とアプリケーションIDを組としたテーブルの一例である。ここでは、ライブラリ部が2つのアプリケーションに資源（ファイル）を提供していることを示す。

【0083】なお、本実施形態においては、各ライブラリ部が提供した資源の資源名とアプリケーションIDを組にしてテーブルを保持することとなるが、カーネル4 4aがテーブルを保持することとしてもよい。また、本実施形態においては、カーネル4 4aがアプリケーションIDを生成することとしているので、複数のミドルウェアを同時に動作させる場合においても、2つのミドルウェアがそれぞれ個別にIDを生成することにより生じるIDの重複を防ぐことができ、カーネル4 4aが正しく資源を管理することができる。

(実施の形態5)図28は、本発明の実施の形態におけるアプリケーション実行箇域100の構成を示す図である。アプリケーション実行箇域100は、アプリケーション/クラスライブラリ記憶部101、アプリケーション管理部102、VM(Virtual Machine)部103、メモリヒープ管理部104、メモリ105を構成する。ここで、メモリヒープ管理部104は、オブジェクト演算結果得部104a、分割ヒープ領域演算部104b、分割ヒープ領域解消部104c、CMD104d、ロック部104e、メモリヒープ領域管理テーブル104fを構成する。

【0084】メモリ105は、複数の分割ヒープ領域1

(18)

特開2002-259146

33

34

分割ヒープ領域は、1つのアプリケーションに対応して扱けられ、対応するアプリケーションに関連するオブジェクトを保持するための領域である。アプリケーション個別に分割ヒープ領域が扱われるのは、1回のがベジコレクションの対象領域をメモリヒープ領域全体ではなく分割ヒープ領域単位にするためである。加えて、アプリケーション終了時にはメモリヒープ領域は分割ヒープ領域単位に解放される。

【0085】アプリケーション/クラスライブラリ記憶部101は複数のアプリケーション及び既存のクラスライブラリを記憶する。アプリケーション管理部102は、外部からアプリケーションの起動及び終了の指示を受け、VM部103に当該アプリケーションの起動及び終了を指示する。また、アプリケーション管理部102は、アプリケーションが終了すると、分割ヒープ領域解放部104cに当該アプリケーションに関連するオブジェクトが記録された分割ヒープ領域の解放を指示する。

【0086】VM部103は、アプリケーション管理部102からアプリケーションの起動指示を受けると、当該アプリケーション用に分割ヒープ領域の割り当てをメモリヒープ管理部104に行なわせ、割り当てられた分割ヒープ領域にアプリケーション/クラスライブラリ記憶部101から当該アプリケーションをロードするとともにロードしたアプリケーションを実行する。

【0087】より詳しく説明すると、アプリケーションのロードは、第1にクラスローダオブジェクトのロード、第2にアプリケーションに関連する個々のオブジェクトのロードに分けられる。まず、クラスローダオブジェクトのロードについて説明する。VM部103は、アプリケーション管理部102から上記アプリケーション起動指示として、当該アプリケーション用のクラスローダオブジェクトの生成指示を受けて、当該クラスローダオブジェクト用のオブジェクト領域の獲得指示(以下、第1獲得指示と呼ぶ)をオブジェクト領域獲得部104aに送出する。ここで、クラスローダオブジェクトとは、対応するアプリケーションに関連するオブジェクトをアプリケーション/クラスライブラリ記憶部101から分割ヒープ領域にロードするためのオブジェクトである。

【0088】VM部103は、第1獲得指示に対応して、オブジェクト領域獲得部104aから、分割ヒープ領域の割り当て及びオブジェクト領域の割り当てを受けると、アプリケーション/クラスライブラリ記憶部101から当該オブジェクト領域にクラスローダオブジェクトをロードして当該クラスローダオブジェクトを実行する。

【0089】次に、アプリケーションに関連する個々のオブジェクトのロードについて説明する。VM部103は、このクラスローダオブジェクトを実行することによ

くタクをアプリケーション/クラスライブラリ記憶部101からオブジェクト領域獲得部104aによって割り当てられたオブジェクト領域にロードする。すなわち、VM部103は、対応するアプリケーションに関連するオブジェクトについてオブジェクト領域の獲得指示(以下、第2獲得指示と呼ぶ)をオブジェクト領域獲得部104aに派出し、第2獲得指示に対する応答として、オブジェクト領域獲得部104aからオブジェクト領域の割り当てを受けると、アプリケーション/クラスライブラリ記憶部101から当該オブジェクト領域にオブジェクトをロードして当該オブジェクトを実行する。上記第2獲得指示は、クラスローダオブジェクト以外のオブジェクトに対するオブジェクト領域の獲得指示であって、実行中のアプリケーションから他のオブジェクトの生成要求があればVM部103から派出される。

【0090】オブジェクト領域獲得部104aは、VM部103からの第1獲得指示を受けると、分割ヒープ領域獲得部104bに分割ヒープ領域の獲得を指示して分割ヒープ領域の割り当てを受けると、当該分割ヒープ領域内にクラスローダオブジェクト領域を獲得し、当該クラスローダオブジェクト領域をVM部103に通知する。

【0091】メモリヒープ領域テーブル104fは、分割ヒープ領域獲得部104bによって獲得された分割ヒープ領域に関する情報(以下分割ヒープ領域情報と呼ぶ)から構成される。図29に示すように各分割ヒープ領域情報は、分割ヒープ領域のID、分割ヒープ領域にロードされるクラスローダオブジェクトのID、分割ヒープ領域の先頭アドレス、分割ヒープ領域全体のサイズ、分割ヒープ領域中の未使用領域を示す自由領域の先頭アドレス、当該分割ヒープ領域中の自由領域のサイズから構成される。メモリヒープ管理テーブル104fへの分割ヒープ領域情報の新規登録は、分割ヒープ領域獲得部104bがオブジェクト領域獲得部104aからの分割ヒープ領域の獲得の指示を受けて分割ヒープ領域を獲得した時に、分割ヒープ領域獲得部104aによって行われる。上記の分割ヒープ領域情報中の自由領域の先頭アドレス及び分割ヒープ領域中の自由領域のサイズに関する情報は、VM部103からの第1又は第2獲得指示によってオブジェクト領域獲得部104aがオブジェクトテーブル104fによって更新される。

【0092】また、オブジェクト領域獲得部104aは、VM部103からの第2獲得指示を受けると、当該第2獲得指示の要求元アプリケーションをロードしたクラスローダオブジェクトを特定し、特定したクラスローダ

(19)

特開2002-259146

35

36

領域管理テーブル104fを参照して特定(例えば特定したクラスローダがCLである場合には、分割ヒープ領域bを特定)し、特定した分割ヒープ領域内でオブジェクト領域の獲得を試みる。オブジェクト領域獲得部104mは特定した分割ヒープ領域からオブジェクト領域を獲得できた場合、当該オブジェクト領域をVM部103に通知し、メモリヒープ管理テーブル104fの当該分割ヒープ領域の情報(当該分割ヒープ領域中の自由領域の先頭アドレス、当該分割ヒープ領域中の自由領域のサイズ)を更新する。

【0093】一方、オブジェクト領域獲得部104aは、当該分割ヒープ領域からオブジェクト領域を獲得できなかった場合は、CC部104dに対し、当該分割ヒープ領域内のガベージコレクションを指示し、ガベージコレクション終了後に再度オブジェクト領域の獲得を試みる。分割ヒープ領域獲得部104aは、オブジェクト領域獲得部104aからの分割ヒープ領域獲得指示を受けると、メモリヒープ領域から分割ヒープ領域を獲得する。分割ヒープ領域獲得部104bは、獲得した分割ヒープ領域104mの情報をクラスローダオブジェクトのIDと組にしてヒープメモリ領域監査テーブル104fに登録する。

【0094】分割ヒープ領域解部104cは、アプリケーション監査部102からアプリケーションの終了を指示されると、アプリケーション監査部102からの分割ヒープ領域の解放指示に応じて、終了したアプリケーションをロードしたクラスローダオブジェクトに対応する分割ヒープ領域をメモリヒープ領域監査テーブル104fから削除し、特定した分割ヒープ領域を解放し、解放した分割ヒープ領域の情報(解放した分割ヒープ領域のID、当該分割ヒープ領域に対応するクラスローダオブジェクトのID、当該分割ヒープ領域の先頭アドレス、当該分割ヒープ領域全体のサイズ、当該分割ヒープ領域中の自由領域の先頭アドレス、当該分割ヒープ領域中の自由領域のサイズ)とそれに対応するクラスローダオブジェクトのID)をメモリヒープ領域監査テーブル104fから削除する。

【0095】CC部104dは、オブジェクト領域獲得部104aの指示に応じて分割ヒープ領域のガベージコレクションを行う。ロック部104gは、CC部104dが分割ヒープ領域のガベージコレクションを行っている間、当該分割ヒープ領域を使用するアプリケーションの実行を停止する。

【0096】図30は、VM部103よりオブジェクト領域獲得指示(第1獲得指示又は第2獲得指示)を受けた後のアプリケーション実行統括100によるオブジェクト領域獲得動作を示すフローチャートを示す。オブジェクト領域獲得部104aは、VM部からのオブジェクト領域獲得指示が第1(クラスローダオブジェクト)獲得指

かを判定する(S1301)。

【0097】第1獲得指示の場合(S1301:Y)、オブジェクト領域獲得部104aは、当該クラスローダオブジェクトに対応する分割ヒープ領域の獲得を分割ヒープ領域獲得部104bに要求し、分割ヒープ領域獲得部104bは、メモリ105から分割ヒープ領域を獲得する(S1302)。分割ヒープ領域獲得部104bは、図29の例に示す通り、メモリヒープ領域監査テーブル104fに当該クラスローダオブジェクトのIDと獲得した分割ヒープ領域の情報を組にして登録する(S1303)。次にオブジェクト領域獲得部104aは、分割ヒープ領域獲得部104bが獲得した分割ヒープ領域内に当該クラスローダオブジェクトのオブジェクト領域を獲得し(S1304)、当該分割ヒープ領域に対応するメモリヒープ領域監査テーブル104fの情報を更新する(S1309)。

【0098】第2獲得指示の場合(S1301:N)、オブジェクト領域獲得部104aは、VM部103から当該オブジェクトが保有するクラスローダIDを取得し(S1304)、メモリヒープ領域監査テーブル104fから上記クラスローダIDに対応する分割ヒープ領域を特定する(S1305)。次に、オブジェクト領域獲得部104aは、特定した分割ヒープ領域内にオブジェクト領域を獲得する(S1306)。オブジェクト領域獲得部104aは、オブジェクト領域の獲得に成功したか否かを判定する(S1307)。

【0099】オブジェクト領域獲得部104aは、オブジェクト領域の獲得に成功した場合(S1307:Y)、当該分割ヒープ領域に対応するメモリヒープ領域監査テーブル104fの情報を更新する(S1309)。オブジェクト領域獲得部104aは、オブジェクト領域の獲得に失敗した場合(S1307:N)、CC部104dに簡単に失敗した分割ヒープ領域内のガベージコレクションを行うことを指示する。CC部104dは指示に応じて当該分割ヒープ領域のガベージコレクション処理を行(S1308)。CC部104dがガベージコレクション処理を行っている間、ロック部104gは、当該分割ヒープ領域を使用するアプリケーションの実行を停止する。オブジェクト領域獲得部104aはガベージコレクション処理後、再度当該分割ヒープ領域からオブジェクト領域を獲得し(S306)。オブジェクト領域の獲得に成功した場合(S1307:Y)、当該分割ヒープ領域に対応するメモリヒープ領域監査テーブル104fの情報を更新する(S1309)。

【0100】このように、アプリケーション実行統括100においては、生成したオブジェクトを配置するためのオブジェクト領域をアプリケーション層に分割したメモリサイズの小さい分割ヒープ領域として割り当てるため、CC部104dによる1回のガベージコレクション処理の負担を軽減することができる。又、VM部103が復

(20)

特開2002-259146

38

104fは分割ヒープ領域のガベージコレクション処理の実行中、当該分割ヒープ領域を使用するアプリケーションだけを停止させるため、VM部103はその他の分割ヒープ領域を使用しているアプリケーションを引き続きた実行させることができる。

【0101】図31は、アプリケーション終了時のアプリケーション実行状態100による分割ヒープ領域の解放の動作を示すフローチャートである。アプリケーション監視部102は、アプリケーションが終了すると、分割ヒープ領域解放部104cに、終了したアプロケーションをロードしたクラスロードオブジェクトに対応する分割ヒープ領域の解放を指示する(S101)。分割ヒープ領域解放部104cは、メモリヒープ領域監視テーブル104fから当該アプロケーションをロードしたクラスロードオブジェクトに対応する分割ヒープ領域を特定し(S102)、その情報を基いて当該分割ヒープ領域を解放する(S103)。分割ヒープ領域解放部104cは、解放した分割ヒープ領域の情報と対応するクラスロードオブジェクトのIDをメモリヒープ領域監視テーブル104fから削除する(S104)。これにより、アプリケーション実行状態100においては、アプリケーション終了する毎に、対応する分割ヒープ領域を一括して、短時間で解放することができる。

【0102】なお、本実施形態においては、分割ヒープ領域獲得部104bは、システムクラスロードオブジェクトを兼ねることなく、分割ヒープ領域をシステムクラスロードオブジェクトの分割ヒープ領域として割当ているが、図32に示すようにメモリ105金体をシステムクラスロード用のメモリ領域(システムヒープ領域105-1)として割当て、その領域内に分割ヒープ領域獲得部104bにて当該のクラスロードオブジェクトに対応する分割ヒープ領域を獲得するようにしてもよい(図32)。この場合、オブジェクトヒープ領域獲得部104aがシステム起動時にシステムクラスロードオブジェクトのオブジェクト領域をシステムヒープ領域105-1内に獲得し、システムクラスロードオブジェクトのIDと領域を接続(接続の便宜上、分割ヒープ領域ID105aと呼ぶ。)の情報(分割ヒープ領域ID(x))、クラスロードID(SCI)、領域先頭アドレス(MA-0)、全体領域サイズ(MS-0)、自由領域先頭アドレス(FA-0)、自由領域サイズ(FS-0))を記してメモリヒープ領域監視テーブル105fに登録する。以後本実施形態の場合と同様に、分割ヒープ領域獲得部104bがシステムヒープ領域105-1から分割ヒープ領域をクラスロードオブジェクト領域に獲得する毎に、分割ヒープ領域獲得部104bは、図32に示すようにメモリヒープ領域監視テーブル104fに当該クラスロードオブジェクトのIDと獲得した分割ヒープ領域の情報を組にして追加し、オブジェクトヒープ領域監視部は図30のフローチャート

れた分割ヒープ領域内にオブジェクト領域を獲得し(S106)、オブジェクト領域の獲得に失敗した場合(S107)、オブジェクトヒープ領域監視部104bは、OD部104dに獲得に失敗した分割ヒープ領域内のガベージコレクションを行うことを指示し、OD部104dは指示に応じて当該分割ヒープ領域内のガベージコレクションを行う(S108)。また、アプリケーションが終了すると、アプリケーション監視部102は、分割ヒープ領域監視部104cに、終了したアプロケーションをロードしたクラスロードオブジェクトに対応する分割ヒープ領域の解放を指示し、分割ヒープ領域監視部104cは、指示に応じて、図31のフローチャートに従って当該分割ヒープ領域を解放する。また、OD部104dはシステムヒープ領域105-1金体をガベージコレクションの対象とし、分割ヒープ領域を単位に解放された分割ヒープ領域回数をまとめるので、システムヒープ領域105-1を効率的に使用することができる。

【0103】なお、本実施形態における図28のアプリケーション実行状態100の構成は、構造ブロック図として示されているが、実際はCPU、メモリ、外部との入出力による出入入力装置を含むハードウェア及びソフトウェアを実行することにより実現される。このソフトウェアは図28に示した各ブロックの機能を管理するよう設計されている。また上記ソフトウェアを、CD-ROMからロードする場合はCD-ROM読み取りインターフェースを用いてもよい。メモリは、RAM、ROM、ハードディスクや、CDドライブ、フロッピーディスク、メモリカードなどのリムーバブルメディアで構成される。

【0104】本実施形態においては、アプリケーションに割当られる分割ヒープ領域の数は、1つに限らず複数であってもよい。例えば、アプリケーション自身がクラスロードを保存している場合には、当該クラスロードに対しても分割ヒープ領域が割当られる。また、本実施形態において、分割ヒープ領域獲得部104bによって獲得される分割ヒープ領域の大きさは、分割ヒープ領域獲得部104bへのプログラム指示により可変とすることができる。

【0105】
【免責の効果】本発明は、アプリケーションに資源を提供する少なくとも1つ以上のライブラリ部とカーネル部を構えるアプリケーション実行装置であって、前記カーネル部は、アプリケーションが終了すると、終了したアプリケーションがどれであるかを資源を提供した各ライブラリ部に通知する通知手段を有し、各ライブラリ部は、通知手段から通知されると、終了したアプロケーションに提供した資源を回収する回収手段を有することを特徴とする。また、上記アプロケーション実行装置における手数をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現

(21)

特開2002-259146

39

40

ションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有することとしてもよいし、前記各ライブラリ部はさらに、アプリケーションからの要求に応じて資源を提供する提供手段と、前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部にアプリケーションが終了すれば前記通知を行うよう依頼する依頼手段とを有することとしてもよい。

【0106】この構成により、アプリケーションが終了する毎に、カーネル部ではなくライブラリ部により不要になった資源が回収されるので、新規の資源が追加される毎にカーネル部を修正する必要がなくなるという効果が得られる。ここで、前記通知手段は、アプリケーション部に前記通知を行う複数の通知手段を有し、前記各ライブラリ部はさらに、アプリケーションからの要求に応じて資源を提供する提供手段と、前記提供手段がアプリケーションに新たな資源の提供を開始したとき、本ライブラリ部に当該アプリケーションが終了すれば前記通知を行なうよう前記通知手段に依頼し、依頼された前記通知手段は、前記当該アプリケーションが終了すると、資源回収インスタンスのメソッドを呼び出すことにより、前記通知を行うこととしてもよい。

【0107】この構成により、終了したアプリケーションに資源を提供しているライブラリ部のみ、終了したアプリケーションがどれであるかが通知されるので、当該資源を提供していないライブラリ部の無駄な通知をすることなく、当該資源を回収することができるという効果が得られる。ここで、前記回収手段は、アプリケーションとアプリケーションに提供した資源との対応関係を示すテーブルを保持するテーブル保持手段と、前記通知手段によって通知されたアプリケーションに提供した資源を前記テーブル保持手段により特定する資源特定手段とを有し、前記通知手段は、複数のアプリケーションが同時に終了した場合に、終了した複数のアプリケーションがどれであるかを、資源を提供したライブラリ部に通知し、前記資源特定手段は、通知された複数のアプリケーションに特定する複数の資源を前記テーブル保持手段により特定し、前記回収手段は、特定された複数の資源を回収することとしてもよい。

【0108】この構成により、ライブラリ部は、終了した複数のアプリケーションに提供した資源を一括して回収ができるので、より効率的な資源の回収を行うことができる。ここで、前記通知手段は、さらにアプリケーショ

ン部に通知し、前記各ライブラリ部はさらに、通知されたアプリケーションが終了したか、中斷したかに応じて、当該アプリケーションに提供した資源を回収するか否かを判定する判定手段を有し、前記回収手段は、さらに資源を回収すると判定された場合に、通知されたアプリケーションに提供した資源を回収することとしてもよい。また、上記アプリケーション実行装置における各手段をコンピュータに実装させるプログラムを記録したコピーライブ読み取り可能な記録媒体として実現する

10 こととしてもよい。

【0109】この構成により、アプリケーションが中断しているか、終了しているかに応じて、資源を回収すべき否かを判定し、判定の一部の資源のみ回収することができる。また、本発明は、アプリケーションに資源を提供するOS部とJavaミドルウェア部を備えるアプリケーション実行装置であって、前記Javaミドルウェア部は、アプリケーションとアプリケーションに対応するタスクとタスクを構成するスレッドの対応関係を示すテーブルを保持する第1テーブル保持手段と、アプリケーション終了の指示を受けると、前記第1テーブル保持手段を参照して終了するアプリケーションに対応するタスクがどれであるかを通知する通知手段とを有し、前記OS部は、アプリケーション部にアプリケーションを実行するタスクを生成するタスク生成手段と、前記タスクを構成するアプリケーション用スレッドを複数生成するスレッド生成手段と、前記アプリケーション用スレッドを実行することにより、アプリケーションのプログラムコードを実行し、アプリケーションが要求する資源を機器し、提供した資源と前記スレッドが属するタスクとの対応関係を示すテーブルを保持する資源手段と、前記通知手段から通知されたタスクに応じる資源を管理手段により特定し、アプリケーションに提供されたいた資源を回収する回収手段とを有することを特徴としてもよい。また、上記アプリケーション実行装置における各手段をコンピュータに実装させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【0110】この構成により、アプリケーションが終了する毎に不要になった資源が回収されるので、Javaアプリケーションを連続して実行するための資源を確保できるという効果が得られる。ここで、前記Javaミドルウェア部はさらに、前記タスクを構成する前記アプリケーション用スレッドとは別に、前記Javaミドルウェア部に必要な資源を確保する資源確保用スレッドを生成する資源確保用スレッド生成手段と、生成した資源確保用スレッドを実行することにより前記Javaミドルウェア部に必要な資源を確保する資源確保手段とを有し、前記回収手段は、前記資源確保手段によって確保された前記Javaミドルウェア部に必要な資源を回収することなく、前記

(22)

特開2002-259146

41

手段より特定し、アプリケーションに提供されていた資源を回収することとしてもよい。

【01111】この構成により、Javaミドルウェア部に必要な資源が確保され、アプリケーションが終了しても当該資源は回収されないので、より短い時間で新たなアプリケーションを実行させることができる。また、本発明はガベージコレクションが必要なアプリケーション用のメモリヒープ領域を管理するアプリケーション実行装置であって、アプリケーションが起動される毎に、メモリヒープ領域から分割ヒープ領域を獲得する分割ヒープ領域を手段とし、起動されたアプリケーションに、前記分割ヒープ領域獲得手段によって獲得された分割ヒープ領域を割り当てた手段と、アプリケーションが終了する毎に、当該アプリケーションに割り当てられた分割ヒープ領域を解放するメモリ解放手段とを備えることを特徴としてもよい。また上記アプリケーション実行装置における各手段をステップとするメモリヒープ管理方法としてもよいし、上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【01112】この構成により、アプリケーションが終了する際に、終了したアプリケーションに割り当てられた分割ヒープ領域が解放され、解放された分割ヒープ領域の内部にはガベージコレクションが必要になるので、ガベージコレクションの負荷を軽減することができる。ここで、前記アプリケーション実行装置はさらに、アプリケーションに開通するオブジェクトに対して、当該アプリケーションに割り当てられた分割ヒープ領域内にオブジェクト編成を導入するオブジェクト編成獲得手段と、前記分割ヒープ領域を対象にガベージコレクションを行うガベージコレクション手段とを備えることとしてもよい。また、上記アプリケーション実行装置における各手段をステップとするメモリヒープ管理方法としてもよいし、上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【01113】この構成により、メモリヒープ領域全体よりもメモリサイズの小さい分割ヒープ領域を対象としてガベージコレクションを行うために、1回のガベージコレクションの負荷を軽減することができる。ここで、前記アプリケーション実行装置はさらに、前記ガベージコレクション手段によって対象となる分割ヒープ領域のガベージコレクションが行われている場合、当該分割ヒープ領域を使用するアプリケーションのみ実行を停止させ、他の分割ヒープ領域を使用するアプリケーションは継続して実行させるロック手段を備えることとしてもよい。また、上記アプリケーション実行装置における各手段を

42

上記アプリケーション実行装置における各手段をコンピュータに実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体として実現することとしてもよい。

【01114】この構成により、ガベージコレクションが行われている間も、当該処理の対象となっている分割ヒープ領域を使用するアプリケーション以外のアプリケーションの実行を継続させてるので、ユーザーは当該処理が終わるのを待つことなく、他のアプリケーションを使つた作業を続行することができる。

10 【図面の簡単な説明】

【符号の説明】

【図1】アプリケーション実行装置10の構成を示すブロック図である。

【図2】コールバック関数を定義したC言語プログラムの一例を示す。

【図3】コールバック関数の登録と呼び出し実行を記述したC言語プログラムの例を示す。

【図4】各ライブラリ部が行う資源提供の処理を示すフローチャートである。

【図5】ファイルシステムを制御するライブラリ部(以下ライブラリ部Fと呼ぶ。)が保持するアプリケーション別子と資源名(ファイル名)とを組にしたテーブルを示す。

【図6】各ライブラリ部が行う資源の回収処理を示すフローチャートである。

【図7】複数のアプリケーションが一齊に終了する場合の各ライブラリ部が行う資源の回収処理のフローチャートを示す。

【図8】アプリケーション実行装置20の構成を示すブロック図である。

【図9】インターフェースを定義するJavaプログラムの例を示す。

【図10】アプリケーション情報インスタンスを生成するクラスの一例を示す。

【図11】資源回収クラスの一例を示す。

【図12】資源回収クラスが生成した資源回収インスタンスの例を示す。

【図13】各ライブラリ部が行う資源提供の処理を示すフローチャートである。

【図14】アプリケーション情報インスタンスがアプリケーションの状態を知るためのメソッドと資源回収インスタンスの登録を解除するメソッドを格納している場合の各ライブラリ部が行う回収の処理を示すフローチャートである。

【図15】アプリケーション実行装置30の構成を示すブロック図である。

【図16】アプリケーションとアプリケーションに対応して生成されたタスクとタスクを構成するスレッドとの

(23)

特開2002-259146

43

- 【図17】アプリケーション終了時にアプリケーション蓄積部3-3が行う処理を示すフローチャートである。
- 【図18】クラスライブラリ部が行うリスナー一覧画面の処理を示すフローチャートである。
- 【図19】クラスライブラリ部が保持するテーブルの例である。
- 【図20】イベントが発生したことを通知する処理を示すフローチャートである。
- 【図21】クラスライブラリ部が行うリスナー呼び出しの処理を示すフローチャートである。
- 【図22】キーを用いて専用スレッドからリスナーを呼び出す手順の例を模式化した図である。
- 【図23】Web部3-3-aがシステム専用のスレッドを監視する処理を示すフローチャートである。
- 【図24】アプリケーション実行装置4-0の構成を示すブロック図である。
- 【図25】アプリケーション蓄積部4-3-bが保持するクラスロードのインスタンスと対応するアプリケーションIDを組にしたテーブルの例を示す。
- 【図26】アプリケーション蓄積部4-3-bが生成したクラスロードのインスタンスとスタックとの関係を示す模式図である。
- 【図27】提供した資源の資源名とアプリケーションIDを組にしたテーブルの一例を示す。
- 【図28】アプリケーション実行装置10-0の構成を示すブロック図である。
- 【図29】メモリヒープ領域管理テーブル10-4-fとメモリ10-5の状態を示す図である。
- 【図30】Web部10-3によりオブジェクト領域獲得指示を受けた後のアプリケーション実行装置10-0によるオブジェクト領域獲得の動作を示すフローチャートである。
- 【図31】アプリケーション終了時のアプリケーション実行装置10-0による分割ヒープ領域の解放の動作を示すフローチャートである。
- 【図32】システムヒープ領域10-5-1がメモリヒープ領域管理テーブル10-4-fに登録されている状態を示す図である。
- 【符号の説明】
- メモリヒープ管理部
 - ヒープ領域管理部
 - CC(カベージコレクション)部
 - アプリケーション実行装置
 - アプリケーション入力部
 - アプリケーション記憶部
 - カーネル部
 - 巡回実行部
 - 資源回収通知部
 - 識別子供送部
 - ライブラリ格納部

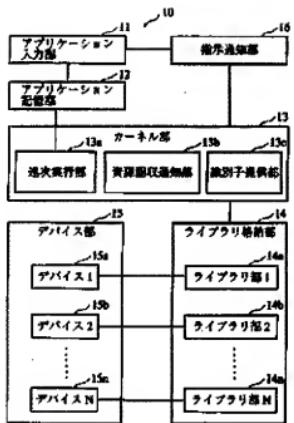
44

- 14-a~n ライブラリ部
15 デバイス部
15-a~n デバイス
20 アプリケーション実行装置
23 OS部
23-a パーチャルマシン部
23-b アプリケーション蓄積部
23-c~e アプリケーション情報インスタンス
24 ライブラリ格納部
10 24-a~n ライブラリ部
30 アプリケーション実行装置
33 Javaミドルウェア部
33-a Web部
33-b アプリケーション蓄積部
33-c クラスライブラリ格納部
33-c1~c2 クラスライブラリ部
34 OS部
34-a カーネル
34-b ライブラリ格納部
20 34-b1~b2 ライブラリ部
35 ハードウェア部
35-a CPU
35-b1~b2 デバイス
40 アプリケーション実行装置
43 Javaミドルウェア部
43-a Web部
43-b アプリケーション蓄積部
43-c クラスライブラリ格納部
43-c1~c2 クラスライブラリ部
30 44 OS部
44-a カーネル
44-b ライブラリ格納部
44-b1~b2 ライブラリ部
45 ハードウェア部
45-a CPU
45-b1~b2 デバイス
100 アプリケーション実行装置
101 アプリケーション/クラスライブラリ記憶部
102 アプリケーション蓄積部
40 103 Web部
104 メモリヒープ蓄積部
104-a オブジェクト領域獲得部
104-b 分割ヒープ領域獲得部
104-c 分割ヒープ領域解放部
104-d QM部
104-e ロック部
104-f メモリヒープ領域管理テーブル
105 メモリ
105-a~105-n 分割ヒープ領域

(24)

特開2002-259146

【図1】



【図2】

```
typedef void(*CALLBACK_f)(int AppId)
```

【図3】

アプリケーション識別子	ファイル識別子
1	aaa
2	bbb

アプリケーション識別子	ファイル識別子
1	ccc
2	ddd

アプリケーション識別子	ファイル識別子
1	eee

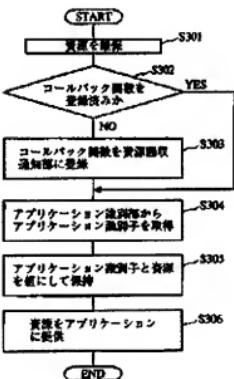
【図4】

【図5】

```

1   CALLBACK_f callf[100];
2
3   void init() {
4       for (l=0; l<100; l++) callf[l]=null;
5   }
6   int add_callf(CALLBACK_f callffunc){
7       for (l=0; l<100; l++){
8           if (callf[l]==null) {
9               callf[l] = callffunc;
10              return(1);
11         }
12     }
13     return(-1);
14   }
15
16   void remove_callf(CALLBACK_f callffunc){
17       for (l=0; l<100; l++){
18           if (callf[l]==callffunc) callf[l] = null;
19       }
20   }
21
22   void call_callback (int AppId) {
23       for(l=0; l<100; l++){
24           if ((callf[l] != null) (*callf[l])(AppId));
25       }
26   }

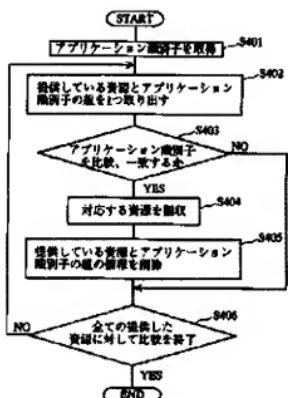
```



(25)

特開2002-259146

【図6】

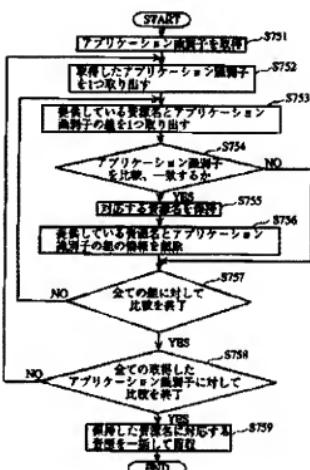


【図9】

```

public interface resourceCollectionListener {
    public void update();
}
    
```

【図7】



【図10】

```

public class applicationProxy {
    public String name; // application name
    public int getStatus() {
        // get the status of application,pause or destroy
    }
    public addListener(resourceCollectionListener l) {
        // register resource collection instance
    }
    public removeListener(resourceCollectionListener l) {
        // remove resource collection instance
    }
}
    
```

【図11】

```

public class fileCollectionListener
    implements resourceCollectionListener {
    String name; // filname
    ApplicationProxy app;
        // Application information instance
    public void update() {
        // release file which has filname
    }
}
    
```

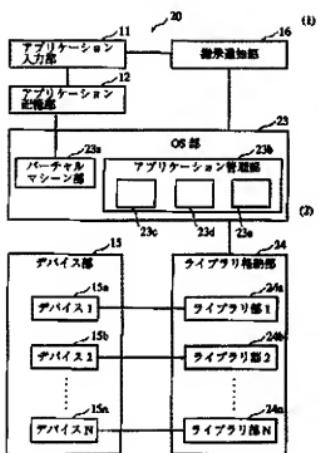
【図25】

アプリケーションID	クラスローダー
1	classloader1
2	classloader2

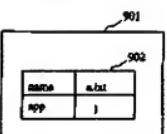
(26)

特開2002-259146

【図8】



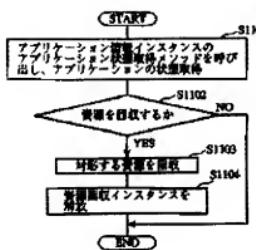
【図12】



【図27】

Application ID	File
1	App1.txt
2	App2.txt

【図14】



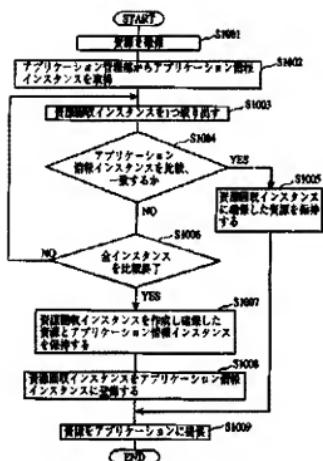
【図16】

Application ID	Task ID	Thread ID	S1101		
			S1102		
1	201	1,2			
2	202	4,5			
Application ID	Task ID	Thread ID	S1103		
			S1104		
1	201	1,2,7			
2	202	4,6			

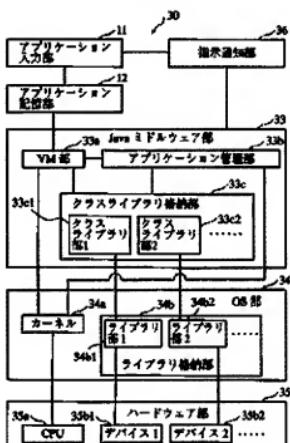
(27)

特開2002-259146

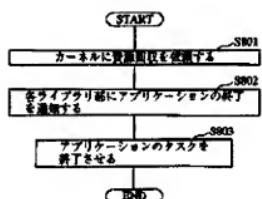
【図13】



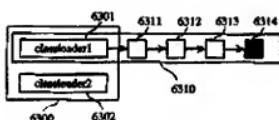
【図15】



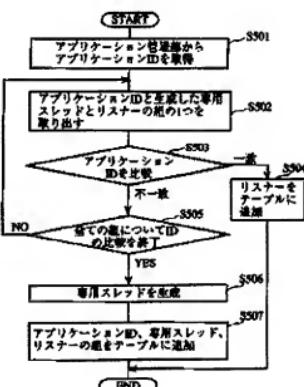
【図17】



【図26】



【図18】



(26)

特開2002-259146

【図19】

(1)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
1	thread10	L1,L4

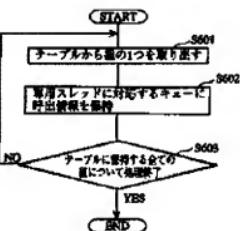
(2)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
1	thread10	L1,L4
2	thread20	L6

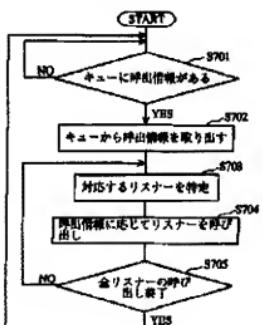
(3)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
1	thread20	L6

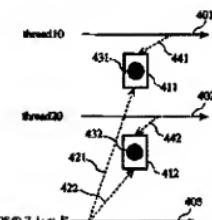
【図20】



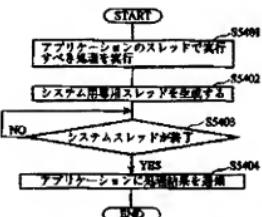
【図21】



【図22】



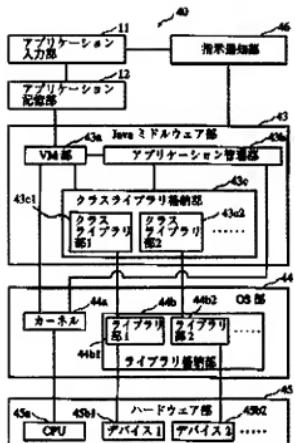
【図23】



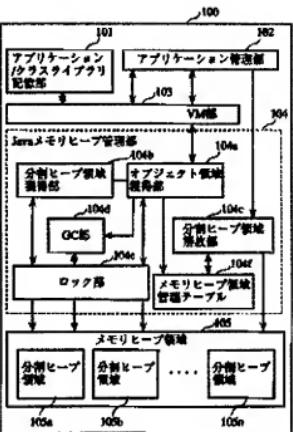
(28)

特開2002-259146

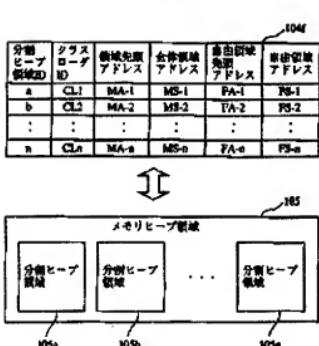
【図24】



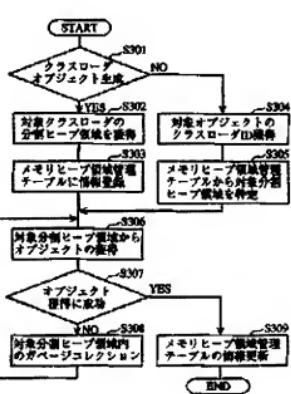
【図28】



【図29】



【図30】



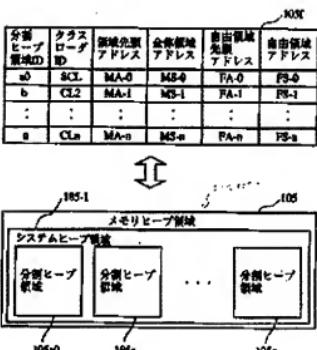
(30)

特開2002-259146

【図31】



【図32】



フロントページの焼き

(72)発明者 平本 雄志
大阪府門真市大字門真1006番地 松下電器
産業 株式会社内

(72)発明者 久保綱 裕子
大阪府門真市大字門真1006番地 松下電器
産業 株式会社内

(72)発明者 土井 翔則
大阪府門真市大字門真1006番地 松下電器
産業 株式会社内

Fターム(参考) 5B060 AA10
5B098 GD03 GD07 GD14 GD22